Indice

Parte I

Introduzione e specifiche

Il documento è suddiviso in 5 parti. Si è cercato di riproporre, un itinerario di lettura che permetta al lettore di ripercorrere le varie fasi del progetto così come sono state affrontate, seguendo come linea di riferimento quella del libro di testo di UML di Arlow e Neustadt [UML]. Il lettore in possesso di una copia del libro di testo consigliato per il corso di Ingegneria del Software avrà quindi più familiarità con lo schema di lettura, che risulta comunque semplice ma nello stesso tempo molto ben dettagliato. Nella prima parte si discutono le tecniche e gli strumenti utilizzati per la gestione del processo e del progetto, il modello di processo scelto e come tale modello è stato personalizzato in base alle nostre esigenze. Questa parte può essere completamente saltata dal lettore che non sia interessato a queste problematiche, il quale può iniziare la lettura direttamente dalla seconda parte senza precludere la comprensione delle specifiche. E' infatti nella seconda parte ("Analisi", cap. 2-8) dove si iniziano a descrivere le specifiche di progetto: il capitolo 2 presenta una breve riflessione su come o cosa dovrebbe fare il sistema; nel capitolo 3 si presenta il glossario dei termini ottenuto da un documento di specifica dei requisiti espanso in linguaggio naturale. Nel capitolo 4 invece si può osservare il documento di descrizione dei casi d'uso, il quale mette in luce i requisiti funzionali del sistema, permette di individuare gli attori principali che interagiranno col sistema e le modalità con le quali tali interazioni avranno luogo. Nel capitolo 5 invece, viene riportata la discussione che ha portato all'individuazione delle classi di analisi, tramite l'utilizzo congiunto dei metodi di analisi grammaticale delle specifiche e CRC. Il risultato è uno scheletro di diagramma delle classi che andrà poi rivisto, ampliato e raffinato in fase di progettazione. I capitoli 6 e 7 presentano rispettivamente le scelte fatte in tema di architettura software (pattern architetturale scelto), e di suddivisione del progetto in packages (e loro disposizione all'interno dell'architettura scelta). Nel capitolo 4 inoltre il lettore potrà prendere visione della realizzazione dei casi d'uso dell'analisi, con la presentazione dei diagrammi di sequenza di alto livello e dei diagrammi di attività per ogni caso d'uso.

La terza parte invece è incentrata sulle attività di progettazione. Il capitolo 8 inizia con l'individuazione di design pattern applicabili al progetto, in modo da tenerne conto nelle fasi successive, ad esempio al momento del raffinamento del diagramma delle classi, cosa che avviene nel capitolo 9. Tale capitolo contiene

anche una spiegazione e una descrizione in linguaggio naturale delle operazioni e degli attributi contenuti in ogni classe. Il capitolo 10 invece fornisce al lettore una realizzazione dei casi d'uso "di progettazione", ovvero ad un livello di dettaglio maggiore: per ogni caso d'uso si presentano il diagramma di sequenza e il diagramma di collaborazione che li realizzano, inoltre viene fornita una spiegazione testuale delle interazioni tra le varie classi necessarie alla realizzazione di ognuno di essi. Il capitolo 11 conclude la parte terza riportando una discussione sui diagrammi degli stati, i criteri per individuare le classi più interessanti riguardo alle quali realizzarli e una spiegazione testuale per ognuno di essi.

Nella quarta parte vengono forniti al lettore alcuni cenni sul deployment della applicazione, utili per farsi un'idea di come saranno organizzati i vari moduli e di come saranno distribuiti sui vari nodi. La lettura di questa parte è facoltativa ai fini della comprensione delle specifiche.

Il documento si conclude con la parte quinta, nella quale vengono fornite alcune considerazioni sui tool utilizzati, eventuali problemi incontrati, informazioni utili al contorno e nella quale vengono tirate le somme dei risultati ottenuti. Il lettore non interessato può saltare questa parte, dato che essa non contiene informazioni strettamente necessarie alla comprensione dello sviluppo del progetto.

Capitolo 1

Gestione del processo

Prima di iniziare la normale attività di analisi e progettazione, abbiamo dedicato alcuni incontri alla pianificazione e gestione del processo e delle risorse. Sono stati affrontati e discussi i seguenti punti, cercando una buona soluzione per ognuno di essi:

- Processo da adottare;
- Pianificazione dei tempi e degli sforzi;
- Adozione di strumenti di ausilio;
- Gestione della risorsa "persone": organizzazione del team e suddivisione dei compiti.

1.1 Processo da adottare

L'ingegneria del software orientata agli oggetti applica un modello di sviluppo che incoraggia lo sviluppo iterativo, per quanto riguarda il processo di gestione pertanto è preferibile la trattazione di una natura evolutiva.

Bernard e Booch in particolare suggeriscono l'uso di un modello ricorsivo/parallelo che proceda iterativamente: i punti principali di questo approccio sono il riconoscere che l'analisi e la progettazione di sistemi Object Oriented non si possono compiere

allo stesso livello di astrazione e che analisi e progettazione si possono applicare in parallelo a componenti indipendenti del sistema.

Per portare avanti l'analisi e la progettazione object-oriented di un servizio web abbiamo scelto di adottare quindi il modello di processo ricorsivo/parallelo (con particolare attenzione alle fasi RUP descritte nel libro [INGSW] [UML]) , in modo da poter supportare meglio i cambiamenti e le correzioni che inevitabilmente si sarebbero rivelate necessarie nel corso dello sviluppo delle specifiche UML. Così facendo si è venuto a creare un "framework" molto flessibile (al contrario di quanto si sarebbe ottenuto con un modello di tipo sequenziale lineare), che ci ha permesso di apportare correzioni e modifiche "retroattivamente", ovvero tornando indietro nelle fasi precedenti ogni qualvolta veniva trovato un errore o una incoerenza. Raramente infatti nei progetti, anche di piccola portata come questo, si riesce ad avere una visione precisa e dettagliata fin da subito dello spazio del problema. Una volta scelto il processo, siamo passati a definire le KPA (Key Process Area), ovvero le attività principali in cui si sarebbe articolato il processo stesso. Abbiamo ritenuto che la suddivisione seguente potesse fare al caso nostro:

- Analisi dei requisiti (KPA1);
    - Stesura delle specifiche espanse in linguaggio naturale;
    - Rimozione delle sinonimie e omonimie;
    - Stesura di un Glossario dei termini; (Milestone 0)
    - Individuazione dei casi d'uso;
    - Stesura del documento descrittivo dei casi d'uso (Milestone 1)
- Analisi (KPA2);
    - Analisi dell'architettura SW da utilizzare;
    - Package di analisi;
    - Individuazione delle classi di analisi: diagramma delle classi di analisi (Milestone 2);
        - Metodo dell'analisi grammaticale

    - Metodo CRC
  - Realizzazione dei casi d'uso dell'analisi:
    Diagramma di sequenza ad alto livello;
  - Diagrammi di attività

- Progettazione (KPA3);
  - Raffinamento dei package di analisi;
  - Diagramma delle classi di progettazione; (Milestone 3)
  - Realizzazione dei casi d'uso di progettazione: (Milestone 4)
    - Diagrammi di sequenza a basso livello
    - Diagrammi di collaborazione
  - Diagrammi degli stati;
  - Individuazione dei design patterns applicabili;

Come attività ausiliarie al processo (e quindi attive durante tutto l'arco dello sviluppo), abbiamo deciso di avvalerci di:

- **Revisioni tecniche formali**: per garantire la qualità, quando un componente del team produce un documento (di qualunque tipo esso sia), alla riunione successiva gli altri 3 membri del team revisionano in maniera indipendente una copia del documento, appuntando errori rilevati e cose poco chiare. Successivamente l'autore del documento viene interpellato per chiarire le cose dubbie e per verificare che i potenziali errori individuati siano effettivamente tali, e non siano dovuti ad una errata comprensione del documento stesso da parte dei revisori. Al termine della revisione, all'autore del documento viene lasciato un altro documento contenente le modifiche da apportare (decise con votazione a maggioranza);

- **Brainstorming**: all'inizio di ogni sottoattività nuova, se necessario, si tiene una riunione dove si parlerà delle tematiche principali inerenti la nuova attività: problemi possibili, dubbi, chiarimenti, idee innovative e altro vengono messe all'ordine del giorno. La riunione è presieduta dal PM, il quale

si occupa di gestirne lo svolgimento evitando confusione e regolando gli interventi. A fine riunione, sempre il PM, assegna a ogni membro del team (lui compreso), dei task (realizzazione di diagrammi, produzione di documenti ecc...) da portare a termine entro la riunione successiva.

- **Punti di controllo dello stato di avanzamento del processo**: si è affidato al PM il compito di controllare lo stato di avanzamento del progetto, in modo da rilevare anticipatamente il prospettarsi all'orizzonte del rischio di forti ritardi. Per fare questo, il PM si è basato sul risultato della pianificazione del progetto come mezzo di paragone rispetto al quale valutare il reale stato del progetto in un certo momento. In caso di ritardi notevoli, il PM dovrà prendere provvedimenti: modificando il piano di progetto o redistribuendo i compiti e motivando i membri del team ad una maggiore produttività, facendo attenzione a non colpevolizzarli.

1.2 Pianificazione dei tempi e degli sforzi

Per portare a termine il progetto abbiamo previsto uno sforzo di circa 60 ore di lavoro, delle quali una buona parte concentrate nelle fasi di individuazione dei requisiti e di analisi, cruciali per una buona riuscita delle attività di progettazione. Nel determinare le deadlines per la consegna delle milestones o dei task assegnati siamo stati il più stringenti possibile, in modo da poter reagire bene ai ritardi che si sarebbero potuti presentare, senza comportare una serie di slittamenti che avrebbero rappresentato un rischio troppo grave, in grado di mettere a repentaglio il successo del progetto stesso.

Ecco una lista delle deadlines principali che ci siamo posti per la produzione dei documenti:

Milestones	Deadline
Glossario dei termini	13/11/03
Descrizione dei casi d'uso	13/11/03
Classi di analisi	21/11/03
Classi di progettazione	27/11/03
Realizzazione dei casi d'uso	10/12/03

L'obiettivo principale, come si può ben intuire anche dalle date fissate come deadlines, è quello di terminare il progetto entro metà dicembre, lasciando il mese successivo a rifiniture e minor-revisions. Questo però non deve assolutamente portare ad una specifica incompleta o mal realizzata (altro rischio individuato durante le riunioni di pianificazione preliminari): infatti è stato deciso all'unanimità che in caso ci si dovesse accorgere di uno slittamento notevole dei tempi (dovuto a imprevisti o problemi inattesi), si decideranno nuove tempistiche. Alla rapidità di sviluppo abbiamo deciso fermamente di anteporre una buona qualità e la correttezza del materiale prodotto.

1.3 Adozione di strumenti di ausilio

Per rendere le comunicazioni più rapide ed efficienti, per velocizzare il lavoro e migliorare la coordinazione, abbiamo deciso di utilizzare un gruppo yahoo (appositamente creato a membership chiusa). I gruppi di yahoo forniscono molti servizi interessanti e che si sono rivelati di grandissima utilità nello svolgimento delle varie fasi del progetto:

- Spazio di 30MB per contenere files condivisi da tutto il gruppo;
- Funzione di agenda, per segnare appuntamenti e meeting;
- Mailing List: lo strumento principale di comunicazione;
- Possibilità di mandare automaticamente un messaggio ai membri del team quando si pubblica un nuovo file nell'area condivisa;
- Database: da noi utilizzato come piccolo database di progetto, utile per tenere traccia delle informazioni essenziali riguardo frequenza delle riunioni e monte-ore speso in ogni momento.
- Sezione Links: vi abbiamo messo tutti i link utili per lo svolgimento del progetto che via via abbiamo trovato sul web.
- Funzione di sondaggio (Poll): è possibile indire un sondaggio a cui i membri del team sono tenuti a rispondere. Utile per prendere decisioni di vario tipo senza doverci necessariamente incontrare.

1.4 Gestione della risorsa “persone”

Abbiamo scelto di organizzare il team in forma democratica: le decisioni sono state prese su votazione a maggioranza e si è scelto di utilizzare una trama di comunicazioni di tipo orizzontale. Nonostante questo si è cercato di lasciare al PM un discreto grado di “controllo” sull’intera attività in modo da impedire un’eccessiva confusione e un rallentamento troppo marcato del lavoro dovuto ad un overhead di comunicazioni eccessivo. Per questo si è scelto di procedere ad una divisione dei compiti abbastanza indipendenti tra i membri del team, in modo da lasciare loro libertà di lavorare in modo autonomo, senza necessariamente andare incontro a stati bloccanti o a necessità di continue comunicazioni tra due o più membri per ottenere chiarimenti. Per la suddivisione dei compiti ci siamo avvalsi di un criterio guidato dagli interessi e capacità personali, in modo da massimizzare la produttività facendo leva sull’interesse personale e sul piacere nel portare avanti task di un certo tipo.

Membro	NickName	Ruolo	Descrizione
Donati Alfredo	Alfi	Project Manager	Cura particolare della leggibilità dei diagrammi prodotti. Attenzione maniacale per i piccoli dettagli estetici. Si occupa di suddividere i compiti e assegnare task ai singoli membri con scadenze di consegna.
Barbieri Luca	Bilbo	Librarian	Spiccato interesse per la progettazione di interfacce web e programmazione web-oriented in generale. Ricopre funzioni di segreteria, tenendo aggiornato il diario di bordo, e si occupa della gestione del sito web.
Cocciarini Marco	Mr. Bad Guy	Technical Promoter	Si occupa principalmente dello studio di nuove tematiche inerenti al progetto, applicazione di concetti studiati, esplorazione di nuove problematiche, correttezza nell'uso della sintassi UML e architettura del SW. Founder e moderatore del Gruppo di discussione yahoo utilizzato dal team come mezzo di supporto.
Palmieri Massimiliano	Gregor	Quality Advisor	Particolare interesse per le questioni di logica di business e verifica di correttezza e qualità del materiale prodotto. Ha curato assieme a Mr.Bad Guy la generazione dei documenti esportandoli in LaTeX, con particolare attenzione alla corretta visualizzazione degli stessi a video e su carta.

Personalità dei membri del team: prima ancora di iniziare le attività di analisi e progettazione, abbiamo indetto un Poll sul nostro gruppo di Yahoo! per analizzare le personalità dei membri del team. è stato chiesto ad ogni membro di descrivere la propria personalità, scegliendo tra una delle seguenti possibilità: task oriented; self oriented.

I risultati della votazione sono stati i seguenti:

Tipo di personalità	**Numero di voti**
Task oriented	2
Self Oriented	1
Interaction Oriented	1
TOTALE	**4**

I risultati ottenuti sono stati utilizzati durante tutto l'arco del progetto dal project manager, allo scopo di evitare situazioni di conflitto o malcontento, di massimizzare il livello di soddisfazione di ogni singolo membro, e di assegnare i compiti nel miglior modo possibile.

1.5 Pianificazione del progetto: stime

Per valutare il tempo necessario alla realizzazione del progetto, abbiamo utilizzato il modello Constructive Cost Model II (COCOMO II). Dopo averne studiato le caratteristiche principali [COCOMO][INGSW], abbiamo scaricato un tool che permette, tramite wizard, di inserire tutti i parametri del progetto che si sta svolgendo. Come risultato, si ottiene una stima degli sforzi necessari per le varie fasi del progetto, espressa per mezzo di tabelle contenute in una serie di report. Confrontando i dati ottenuti con il diagramma di Gantt delle attività svolte, si può notare che la stima complessiva del modello COCOMO II è stata abbastanza accurata: infatti è stato previsto uno sforzo complessivo di 3,7 mesi togliendo dai quali le fasi di construction (CN) e transition (TR) non previste dal nostro progetto, si ottiene uno sforzo stimato di 1,5 mesi.

IngSW - Detail Report

Costar 7.0 Demo 05/01/2004 15.08.43 Page: 1

Estimate Name:	IngSW	Estimate ID:	
Model Name:	COCOMO II 2000	Model ID:	2000
Process Model:	COCOMO II Model	Phases:	MBASE

Component Name:	Component1	Component ID:	
Increment:	1	Level:	1
Developed Size:	600	EAF:	0.3418

Phase	Effort (Person-Months)	Cost (K$)	Duration (Months)	Staffing
IN -- Inception	0.0	0.0	0.4	0.1
EL -- Elaboration	0.1	0.0	1.1	0.1
CN -- Construction	0.4	0.0	1.9	0.2
Development (EL+CN)	0.6	0.0	3.1	
TR -- Transition	0.1	0.0	0.4	0.2
Totals (IN+EL+CN+TR)	0.7	0.0	3.8	
MN -- Maintenance (per year)	0.0	0.0		0.0

Figura 1.1: Prima stima

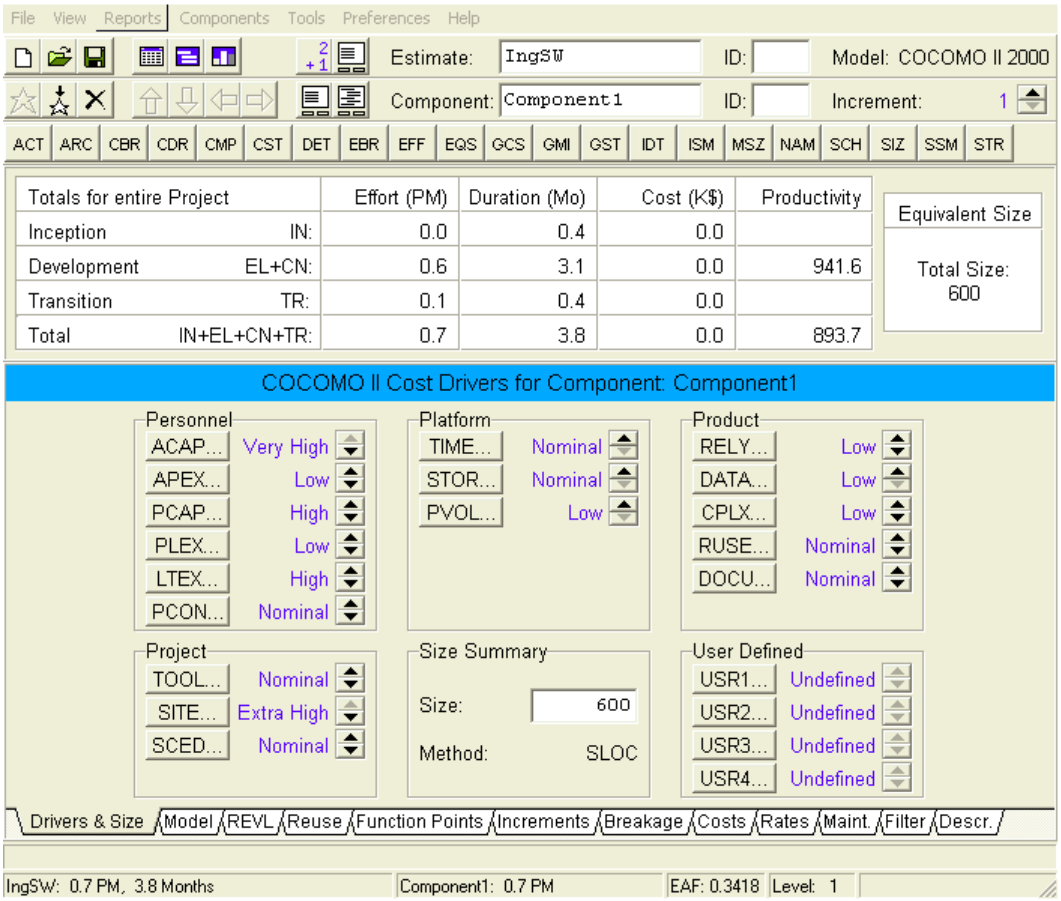


Figura 1.2: Seconda Stima

IngSW - Activity Report

Costar 7.0 Demo 05/01/2004 15.08.43 Page: 1

Estimate Name:	IngSW	Estimate ID:	
Model Name:	COCOMO II 2000	Model ID:	2000
Process Model:	COCOMO II Model	Phases:	MBASE

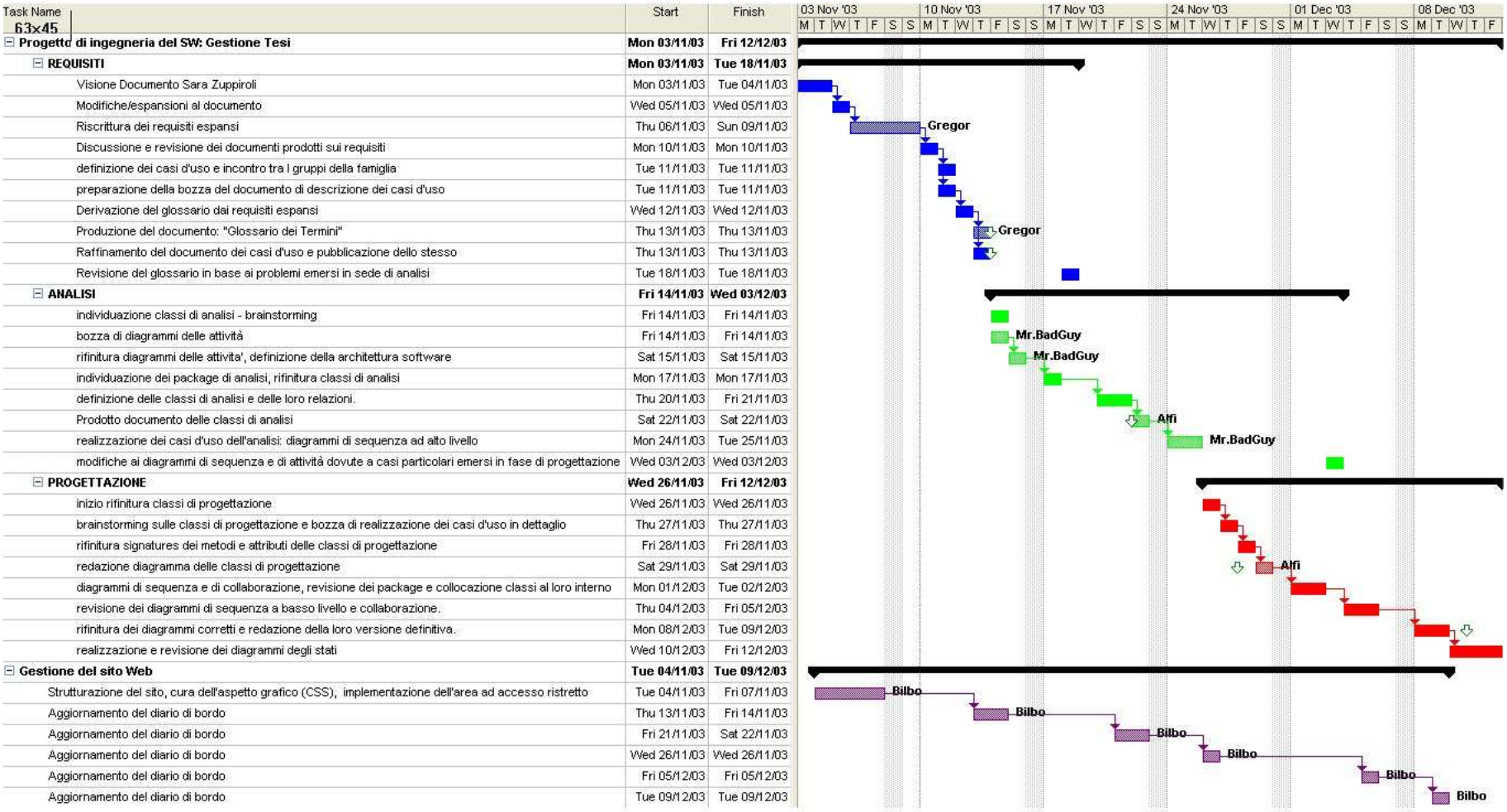
	Effort in Person-Months					
Activity	IN	EL	CN	TR	Total IN to TR	MN
Management	0.0	0.0	0.0	0.0	0.1	0.0
Environment / CM	0.0	0.0	0.0	0.0	0.0	0.0
Requirements	0.0	0.0	0.0	0.0	0.1	0.0
Design	0.0	0.0	0.1	0.0	0.1	0.0
Implementation	0.0	0.0	0.1	0.0	0.2	0.0
Assessment	0.0	0.0	0.1	0.0	0.1	0.0
Deployment	0.0	0.0	0.0	0.0	0.0	0.0
Totals	0.0	0.1	0.4	0.1	0.7	0.0

Figura 1.3: Terza Stima

1.6 Diagramma di Gantt del processo di sviluppo

Come si evince chiaramente dal diagramma di Gantt delle attività svolte, su cinque deadlines fissate, due sono state portate a termine proprio allo scadere del termine ultimo. Le successive due attività, Documento delle classi di analisi e diagramma delle classi di progettazione, sono state portate a termine rispettivamente con uno e due giorni di ritardo rispetto alla deadline fissata. Accortosi del trend negativo, il PM ha indetto una breve riunione dove ha chiesto a tutti i membri del team un impegno maggiore in termini di tempo e risorse. Nel corso della riunione si è tentato di chiarire i motivi dei ritardi, che sono stati individuati in una difficoltà di comprensione degli schemi di comunicazioni tra classi e delle funzionalità da assegnare ad ogni classe, allo scopo di renderle il più possibile coese e disaccoppiate ma nello stesso tempo evitando di sovraccaricarle di responsabilità (regola del "sette più o meno due"). Una volta individuato il problema, è stato allocato del tempo ai chiarimenti del caso e alle contromisure necessarie ad evitare futuri slittamenti. Si è deciso di aumentare l'impegno personale di ogni membro del

team senza esagerare, inoltre è stato aumentato il grado di controllo del PM sugli altri membri del team e si è tentato di innalzare il livello di coesione del team, motivando maggiormente ogni singolo membro e migliorando l'ambiente di lavoro, spostando la sede degli incontri dal laboratorio (ambiente troppo affollato, caldo e rumoroso) all'abitazione privata di uno dei membri del team. Come si può notare anche dal diagramma di Gantt, le contromisure hanno avuto un ottimo effetto, portando addirittura ad un completamento del task "redazione dei diagrammi di sequenza a basso livello e di collaborazione" con tre giorni di anticipo rispetto alla deadline fissata.



Task Name	Start	Finish
Progetto di ingegneria del SW: Gestione Tesi	**Mon 03/11/03**	**Fri 12/12/03**
REQUISITI	**Mon 03/11/03**	**Tue 18/11/03**
Visione Documento Sara Zuppiroli	Mon 03/11/03	Tue 04/11/03
Modifiche/espansioni al documento	Wed 05/11/03	Wed 05/11/03
Riscrittura dei requisiti espansi	Thu 06/11/03	Sun 09/11/03
Discussione e revisione dei documenti prodotti sui requisiti	Mon 10/11/03	Mon 10/11/03
definizione dei casi d'uso e incontro tra I gruppi della famiglia	Tue 11/11/03	Tue 11/11/03
preparazione della bozza del documento di descrizione dei casi d'uso	Tue 11/11/03	Tue 11/11/03
Derivazione del glossario dai requisiti espansi	Wed 12/11/03	Wed 12/11/03
Produzione del documento: "Glossario dei Termini"	Thu 13/11/03	Thu 13/11/03
Raffinamento del documento dei casi d'uso e pubblicazione dello stesso	Thu 13/11/03	Thu 13/11/03
Revisione del glossario in base ai problemi emersi in sede di analisi	Tue 18/11/03	Tue 18/11/03
ANALISI	**Fri 14/11/03**	**Wed 03/12/03**
individuazione classi di analisi - brainstorming	Fri 14/11/03	Fri 14/11/03
bozza di diagrammi delle attività	Fri 14/11/03	Fri 14/11/03
rifinitura diagrammi delle attivita', definizione della architettura software	Sat 15/11/03	Sat 15/11/03
individuazione dei package di analisi, rifinitura classi di analisi	Mon 17/11/03	Mon 17/11/03
definizione delle classi di analisi e delle loro relazioni.	Thu 20/11/03	Fri 21/11/03
Prodotto documento delle classi di analisi	Sat 22/11/03	Sat 22/11/03
realizzazione dei casi d'uso dell'analisi: diagrammi di sequenza ad alto livello	Mon 24/11/03	Tue 25/11/03
modifiche ai diagrammi di sequenza e di attività dovute a casi particolari emersi in fase di progettazione	Wed 03/12/03	Wed 03/12/03
PROGETTAZIONE	**Wed 26/11/03**	**Fri 12/12/03**
inizio rifinitura classi di progettazione	Wed 26/11/03	Wed 26/11/03
brainstorming sulle classi di progettazione e bozza di realizzazione dei casi d'uso in dettaglio	Thu 27/11/03	Thu 27/11/03
rifinitura signatures dei metodi e attributi delle classi di progettazione	Fri 28/11/03	Fri 28/11/03
redazione diagramma delle classi di progettazione	Sat 29/11/03	Sat 29/11/03
diagrammi di sequenza e di collaborazione, revisione dei package e collocazione classi al loro interno	Mon 01/12/03	Tue 02/12/03
revisione dei diagrammi di sequenza a basso livello e collaborazione.	Thu 04/12/03	Fri 05/12/03
rifinitura dei diagrammi corretti e redazione della loro versione definitiva.	Mon 08/12/03	Tue 09/12/03
realizzazione e revisione dei diagrammi degli stati	Wed 10/12/03	Fri 12/12/03
Gestione del sito Web	**Tue 04/11/03**	**Tue 09/12/03**
Strutturazione del sito, cura dell'aspetto grafico (CSS), implementazione dell'area ad accesso ristretto	Tue 04/11/03	Fri 07/11/03
Aggiornamento del diario di bordo	Thu 13/11/03	Fri 14/11/03
Aggiornamento del diario di bordo	Fri 21/11/03	Sat 22/11/03
Aggiornamento del diario di bordo	Wed 26/11/03	Wed 26/11/03
Aggiornamento del diario di bordo	Fri 05/12/03	Fri 05/12/03
Aggiornamento del diario di bordo	Tue 09/12/03	Tue 09/12/03

Legenda:

- = attività di Analisi dei Requisiti svolta da tutto il guppo
- = attività di Analisi dei Requisiti svolta da un singolo membro
- = attività di Analisi svolta da tutto il gruppo
- = attività di Analisi svolta da un singolo membro
- = attività di Progettazione svolta da tutto il gruppo
- = attività di Progettazione svolta da un singolo membro
- = attività ausiliaria (WEB)
- = Deadline

Figura 1.4: Diagramma di Gantt

1.7 Utilizzo della risorsa on-line YahooGroups

Come già detto in precedenza, la maggior parte delle comunicazioni e delle discussioni non affrontabili durante gli incontri sono state registrate presso un gruppo Yahoo creato appositamente. E' interessante notare la partecipazione dei vari membri, in termini di messaggi inviati, nel grafico a torta sottostante.

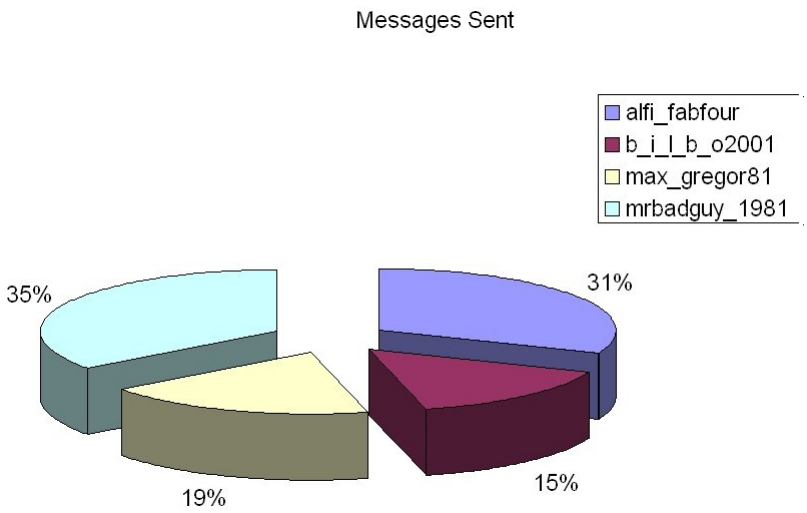


Figura 1.5: Grafico a torta dei messaggi spediti

La maggior parte dei messaggi proviene dal responsabile del gruppo Yahoo (35%), cioè dall'utente forse più attivo del gruppo, il 31% dei messaggi pubblicati e registrati sono invece del Project Manager. Come ovviamente ci si aspetta, proprio il ruolo di coordinatore e moderatore ha richiesto una forte presenza durante la fase di controllo e sviluppo; per quanto riguarda gli altri due membri del gruppo, si nota una sostanziale coerenza con i rispettivi compiti. Il controllore della qualità e il gestore del sito, come si può vedere dal grafico, hanno partecipato attivamente alle discussioni anche se con disponibilità e ruoli subordinati rispetto al Project Manager e al responsabile tecnico.

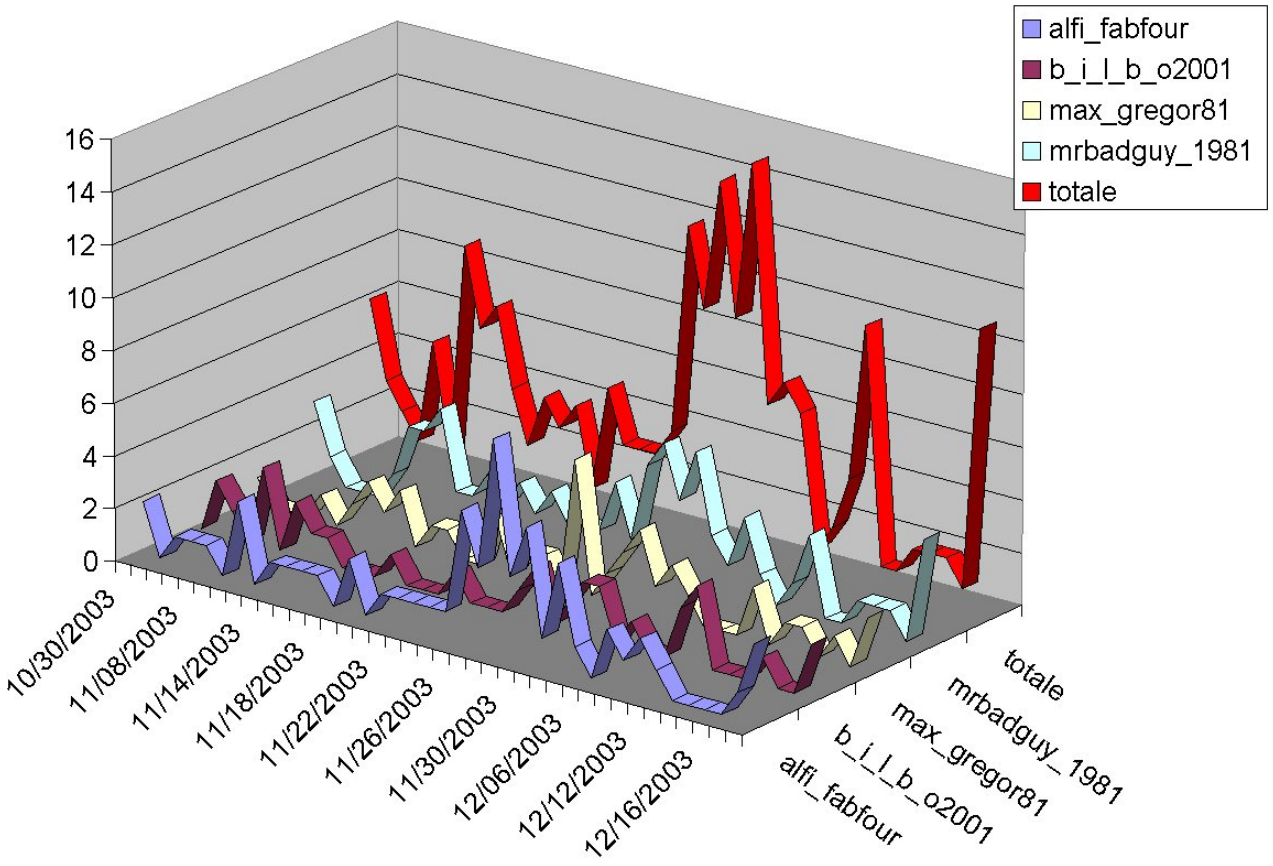


Figura 1.6: Grafico a nastri dei messaggi spediti

E' interessante analizzare il grafico a nastro precedente, in cui si nota l'andamento e l'intensità dello scambio di opinioni e files tramite il gruppo Yahoo, e confrontarlo con il diagramma di Gantt precedentemente riportato. Il primo picco è riscontrabile nel periodo attorno alla metà di novembre, in concomitanza con

l'esigenza di rispettare le deadlines per la pubblicazione del Glossario dei Termini e del documento, comprensivo di diagrammi, di descrizione dei casi d'uso.

Nella settimana successiva si può riscontrare un trend positivo, ed un incremento del numero di messaggi inviati al fine di produrre in tempo la documentazione relativa alle classi di analisi. Dal grafico emerge un sostanziale impegno da tutti i componenti del gruppo con lo scopo di chiarire dubbi, fare nuove proposte e sottoporle al vaglio degli altri componenti del team. E' stato necessario un certo overhead di comunicazioni, data l'impossibilità di vederci fisicamente in quel periodo, per cui la produttività è leggermente scesa e ciò ci ha portato ad un lieve ritardo rispetto alla deadline prefissata.

Il periodo di tempo tra la terza e la quarta settimana di Novembre, fino ai primi giorni di Dicembre, è caratterizzato da un picco di intensità massima in termini di scambio di messaggi e produttività. Confrontando accuratamente i dati riportati nel grafico a nastri con l'andamento del grafico di Gantt e le deadlines prefissate, ci si accorge che tale periodo coincide con la fase di creazione,svilippo e primo raffinamento del diagramma della classi di progettazione. Il motivo di una mole tanto spropositata di messaggi scambiati nonostante i frequenti incontri, è dovuta alla scoperta di errori e a continui raffinamenti dei documenti in fase di produzione. Lo scambio di vedute avvenuto tramite mailing-list ha infatti portato, come si evince chiaramente dal diagramma di Gantt, a un ritorno del progetto alla fase di Analisi, dove sono stati effettuati alcuni aggiustamenti di errori venuti alla luce solo in fase di progettazione. La modifica più significativa e controversa (la sua approvazione ha richiesto un lungo dibattito sia nella mailing-list sia negli incontri dal vivo) ha riguardato il ruolo dei metadati e la loro rappresentazione come classe di analisi. I risultati di questa discussione possono essere presi in visione saltando ai capitoli 4 e 10.

Dopo aver risolto questa questione, il lavoro è proceduto abbastanza regolarmente, con un notevole impegno da parte di tutti i membri al fine di rispettare il più possibile le deadlines successive. Per produrre il diagramma delle classi di progettazione, ad esempio, si è partiti subito con la realizzazione di versioni preliminari dello stesso. Le versioni successive venivano redatte a seguito di modifiche (sostanziali o semplicemente volte a migliorare la leggibilità) proposte dai vari membri del team nel corso degli incontri solo se approvate da tutti gli altri.

Parte II

Analisi

Capitolo 2

Introduzione: come e cosa vuole fare il sistema

Ciò che il nostro sistema intende realizzare è un servizio per la gestione delle tesi in ambito accademico. Il lavoro di analisi e progettazione è stato assegnato a 4 gruppi indipendenti di 4-5 membri ognuno. La stesura della relazione, le decisioni progettuali e le scelte di modellazione sono state effettuate autonomamente da ogni singolo gruppo, sebbene siano stati necessari un paio di incontri per stabilire una linea di sviluppo comune. Il lavoro dei diversi team si dirama differentemente a seconda delle decisioni e delle politiche di realizzazione interne, senza però poter prescindere da regole e specifiche comuni. Tali specifiche, ampiamente discusse, contestate e approvate hanno prodotto un quadro generale di ciò che "il progetto avrebbe dovuto fare", o meglio, ciò che "il nostro elaborato avrebbe dovuto modellare". Abbiamo scelto un iter comune, un percorso, attraverso la descrizione piuttosto specifica degli scenari più rilevanti senza pregiudicare la libertà di personalizzazione e differenziazione in termini di contenuti e qualità . Dei tanti scenari possibili, abbiamo deciso di modellare le seguenti operazioni:

- Inserimento, modifica ed eliminazione da parte di un docente di una proposta di tesi di laurea in una bacheca on line;
- Scelta da parte dello studente di una proposta di tesi ed eventuale richiesta di assegnazione verso il docente opportuno;

- Assegnazione definitva di una proposta di tesi ad uno studente;
- Consegna in segreteria della copia cartacea (con allegata la versione elettronica) della tesi finita prima della discussione davanti alla commissione;
- Consultazione dell'archivio centrale della segreteria per ottenere copie degli elaborati di tesi.

Ogni gruppo deve obbligatoriamente modellare e sviluppare documentazione relativa esclusivamente ai 7 scenari proposti. In questo modo i progetti saranno confrontabili in termini di accuratezza, livello di dettaglio e correttezza formale. Sebbene implicite, non sono state definite politiche di autenticazione o sicurezza, nè sono stati definiti comportamenti standard in caso di errore. L'unica difficoltà di incontro ha riguardato il comportamento del sistema nel caso di scelta, da parte dello studente, della proposta di tesi. Alla fine abbiamo raggiunto un accordo, decidendo, seguendo una logica interna che forse differisce dalla normale procedura burocratica, secondo cui la scelta di una proposta di tesi non vincola il professore nell'assegnazione: sono due casi d'uso distinti la scelta e l'assegnazione della tesi. La scelta effettuata dello studente ha idealmente il compito di "prenotare" un ricevimento con il docente interessato (e nella nostra implementazione) fornendo a quest'ultimo l'indirizzo email dello studente per ulteriori contatti o conferme. Il professore è tutelato dal sistema, perchè quest'ultimo garantisce che le richieste di tesi provengano solo da studenti con un numero di crediti necessario per sostenere il tipo di tesi richiesto. Allo stesso modo, lo studente non deve necessariamente conoscere i dettagli e la forma della richiesta ma solo attendere una comunicazione da parte del docente interessato.

Dopo aver raccolto un glossario ed un insieme delle frasi relative ai soggetti/oggetti interessati dalla modellizzazione, viene presentata una rassegna descrittiva a parole, integrata da diagrammi, via via sempre più dettagliati dei casi d'uso specifici. La tecnica delle frasi relative alle entità coinvolte è già stata utilizzata in elaborati di altri corsi che hanno richiesto modellizzazioni "secondo specifiche pseudo reali ma generiche" [BD].

Capitolo 3

Glossario dei termini

Frasi relative alla Prova Finale:

Ogni tesi ha un titolo, generalmente univoco per ogni sessione di laurea, proposto da un professore.

Il professore Universitario propone tesi limitatamente all'ambito dei propri corsi.

La tesi può essere di natura compilativa, di ricerca o di rassegna ed è assegnata ad un solo studente.

Ad ogni prova finale discussa viene assegnato un punteggio a seconda del tipo e della qualità.

Le tesi vengono archiviate al momento della presentazione in segreteria e sono identificabili univocamente dalla tripla <data di discussione, nome_studente, titolo_tesi>, infatti per ogni sessione di laurea due studenti omonimi non possono condividere il medesimo titolo di laurea (cosa che potrebbe accadere per appelli successivi).

Frasi relative allo studente:

Ogni studente può fare "domanda di laurea" ad uno o più professori richiedendo l'assegnazione di un solo titolo di tesi. Lo studente può rifiutare le proposte di un professore dopo una breve discussione sull'impegno e il contenuto di una tesi.

Uno studente può decidere di abbandonare/cambiare il progetto di tesi prima della consegna solo per validi motivi e obbligatoriamente dopo una discussione col suo relatore.

Uno studente può proporre eventualmente un titolo di tesi al professore.

Uno studente deve possedere 150 crediti per porgere domanda di laurea.

Frasi relative ad un professore:

Un professore è libero di proporre più titoli di tesi inerenti ai proprio corsi e al corso di laurea cui appartiene.

Un professore non può assegnare più di un titolo di tesi a ciascuno studente, in ogni caso l'assegnazione è vincolata dal possesso di 150 crediti da parte del tesista.

Un docente ha la facoltà di modificare, aggiornare e cancellare l'elenco dei titoli delle tesi disponibili e, solo in casi straordinari, revocare o modificare una tesi già assegnata.

Un professore non può assegnare per qualunque motivo due tesi con lo stesso nome a studenti diversi nella medesima sessione di laurea.

Frasi relative ai crediti:

Ogni esame ha associato un numero di crediti proporzionale alla difficoltà e all'impegno richiesto.

I crediti di ogni esame vengono sommati mano a mano che lo studente supera con esito positivo la prova di fine corso.

I crediti si distinguono a seconda dell'ambito della materia cui sono associati: nel nostro caso esistono crediti informatici, fisici, matematici.

La natura dei crediti è ininfluente riguardo le considerazioni fatte sin ora: uno studente deve possedere 150 crediti totali prima di poter vedersi assegnato un titolo di tesi.

Frasi relative ai metadati:

I metadati sono informazioni aggiuntive riguardanti le proposte di tesi e gli elaborati finali. Obbligatoriamente ogni proposta di tesi deve comprendere informazioni come il docente proponente, la materia relativa ad essa e un titolo; nel caso delle tesi assegnate è necessario specificare anche il tesista che la sta preparando. Una tesi contiene informazioni primarie, come quelle appena descritte, e informazioni al "corredo" non fondamentali per la descrizione della tesi stessa, definite o durante la procedura di pubblicazione della proposta, o durante la procedura di assegnazione o, eventualmente, al momento della consegna della tesi svolta in segreteria.

Le informazioni aggiuntive includono: il tipo di laurea per cui l'elaborato di tesi è progettato (nuovo ordinamento, vecchio ordinamento o triennale), il tipo di lavoro richiesto (implementazione, sintesi di altro materiale, rassegna o ricerca), la sessione di laurea in cui si intende discutere la tesi, la lingua in cui tale tesi è scritta, la data di consegna in segreteria, una breve descrizione del contenuto (abstract) e cinque parole chiave utili al fine di una catalogazione per una ricerca tematica.

Frasi relative agli impiegati:

Gli impiegati sono particolari utenti del sistema che hanno il compito di interagire con lo studente supportando la fase di consegna dell'elaborato finale in vista di una successiva archiviazione. Gli impiegati della segreteria prendono in consegna sia la copia cartacea che la copia in formato elettronico della tesi svolta registrando tale operazione e, se tutte le regole e le scadenze sono state rispettate correttamente, dando possibilità allo studente di accedere alla discussione di laurea.

Capitolo 4

Descrizione dei casi d’uso

Casi d’uso individuati:

1. Inserimento di una nuova proposta di tesi.
2. Modifica di una proposta di tesi esistente.
3. Eliminazione di una proposta di tesi.
4. Scelta di una tesi.
5. Assegnazione di una tesi.
6. Consegna ed archiviazione della prova finale in segreteria.
7. Consultazione dell’archivio delle tesi.

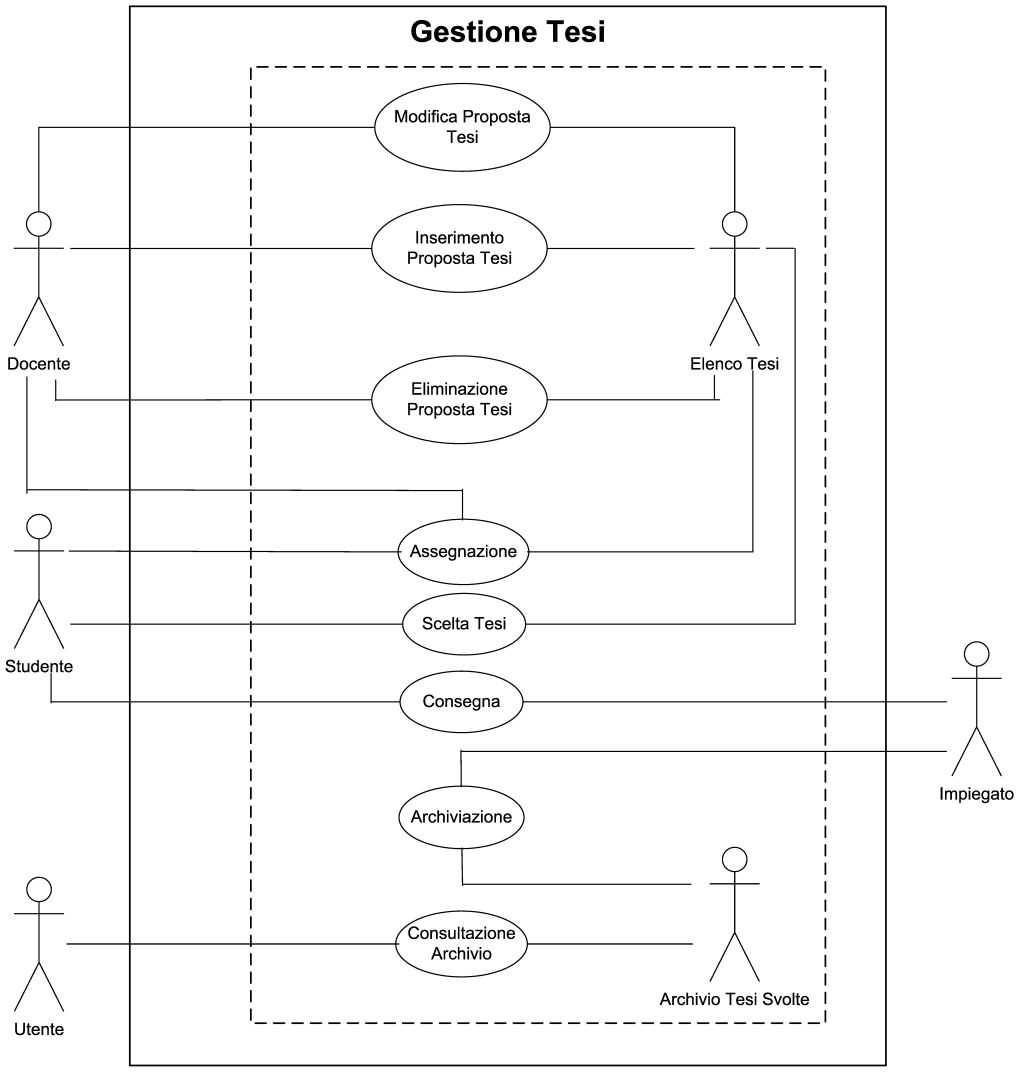


Figura 4.1: Visione completa dei casi d'uso

4.1 Caso d'uso 1: Inserimento di una nuova proposta di tesi

Dichiarazione degli obiettivi: un docente vuole inserire una nuova proposta di tesi nell'elenco delle tesi disponibili.

Descrizione

a. il docente si identifica al sistema;

b. il docente inserisce i dati relativi alla nuova proposta di tesi;

c. il sistema controlla la correttezza dei dati inseriti;

d. il sistema inserisce i dati relativi alla nuova prova finale nell'elenco delle tesi disponibili

Alternativa: *il docente non viene riconosciuto dal sistema.*

Al passo **a**, l'autenticazione al sistema ha esito negativo.

Il sistema chiede nuovamente al docente di autenticarsi.

Alternativa: *i dati inseriti sono scorretti e/o incompleti.*

Al passo **c**, il sistema rileva che i dati inseriti non sono corretti e/o che non sono stati inseriti tutti i dati necessari (Esiste già una proposta di tesi con lo stesso titolo oppure la materia associata alla proposta di tesi non è di competenza del docente).

Il sistema chiede di correggere e/o completare i dati inseriti.

Priorità: alta

Ipotesi: Il sistema possiede un meccanismo di autenticazione per i docenti.

Precondizioni: Il docente ha intenzione di pubblicare una nuova proposta di tesi disponibile.

Postcondizioni: La nuova proposta di tesi è presente nell'elenco delle tesi disponibili.

Problemi aperti: –

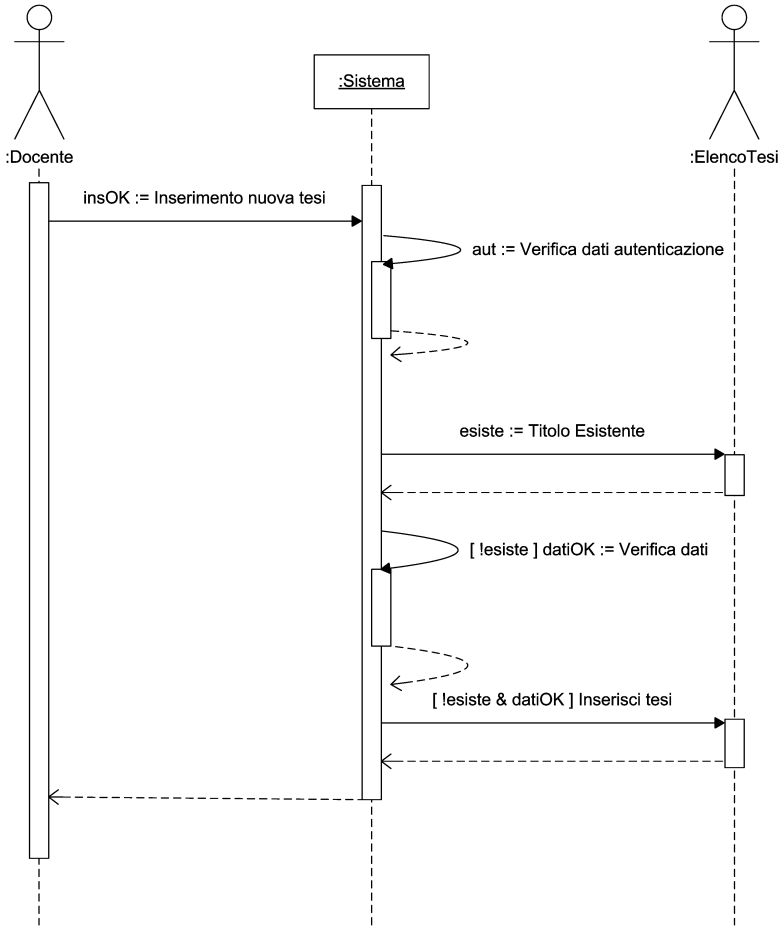


Figura 4.2: Diagramma di sequenza ad alto livello

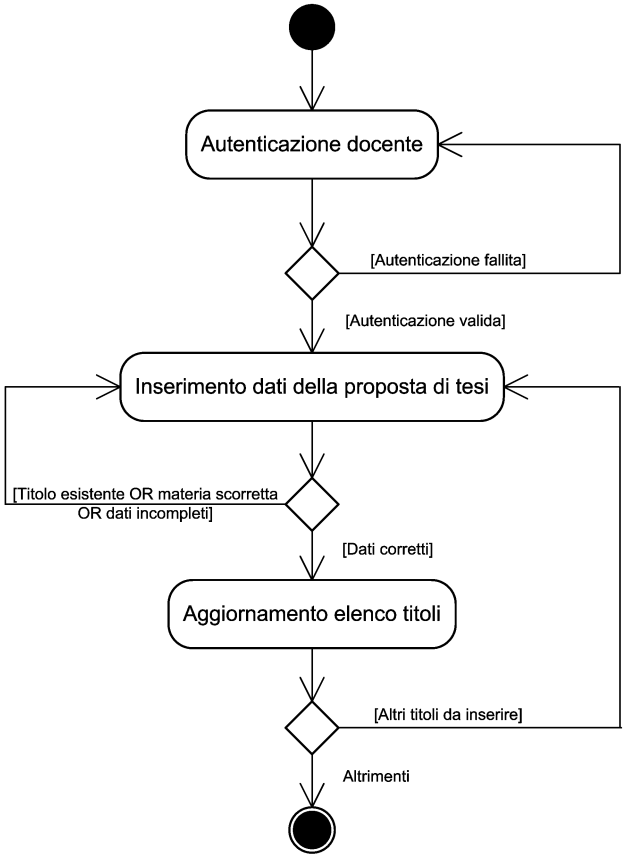


Figura 4.3: Diagramma di attività

4.2 Caso d'uso 2: Modifica di una proposta di tesi esistente

Dichiarazione degli obiettivi: un docente vuole modificare una proposta di tesi precedentemente inserita nell'elenco delle tesi disponibili.

Descrizione:

a. il docente si identifica al sistema;

b. il docente indica quale proposta di tesi modificare e i nuovi dati ad essa relativi;

c. il sistema controlla i dati inseriti;

d. il sistema modifica la proposta di tesi in base ai dati forniti.

Alternativa: *il docente non viene riconosciuto dal sistema.*

Al passo **a**, l'autenticazione al sistema ha esito negativo.

Il sistema chiede nuovamente al docente di autenticarsi.

Alternativa: *il docente richiede di modificare una proposta di tesi inesistente o una da lui non proposta.*

Al passo **c** il sistema rileva che la tesi da modificare non esiste o non è di competenza del docente.

Il sistema notifica l'immissione di dati errati e permette di reinserirli.

Alternativa: *i dati inseriti non sono corretti.*

Al passo **c** il sistema rileva che i dati inseriti non sono corretti (Esiste già una proposta di tesi con lo stesso titolo o la materia associata alla nuova proposta di tesi non è di competenza del docente).

Il sistema notifica l'immissione di dati errati e permette di reinserirli.

Priorità: media

Ipotesi: Il sistema ha un elenco dei docenti, delle tesi disponibili, e delle tesi assegnate. Tali elenchi tengono traccia della corrispondenza tra le tesi e i docenti che le hanno proposte e dello stato della tesi: cioè se è stata assegnata o sia ancora disponibile.

Precondizioni: Il docente ha intenzione di modificare una proposta di tesi da lui precedentemente proposta.

Postcondizioni: Le informazioni relative al titolo da modificare sono state aggiornate in base ai dati forniti dal docente.

Problemi aperti: –

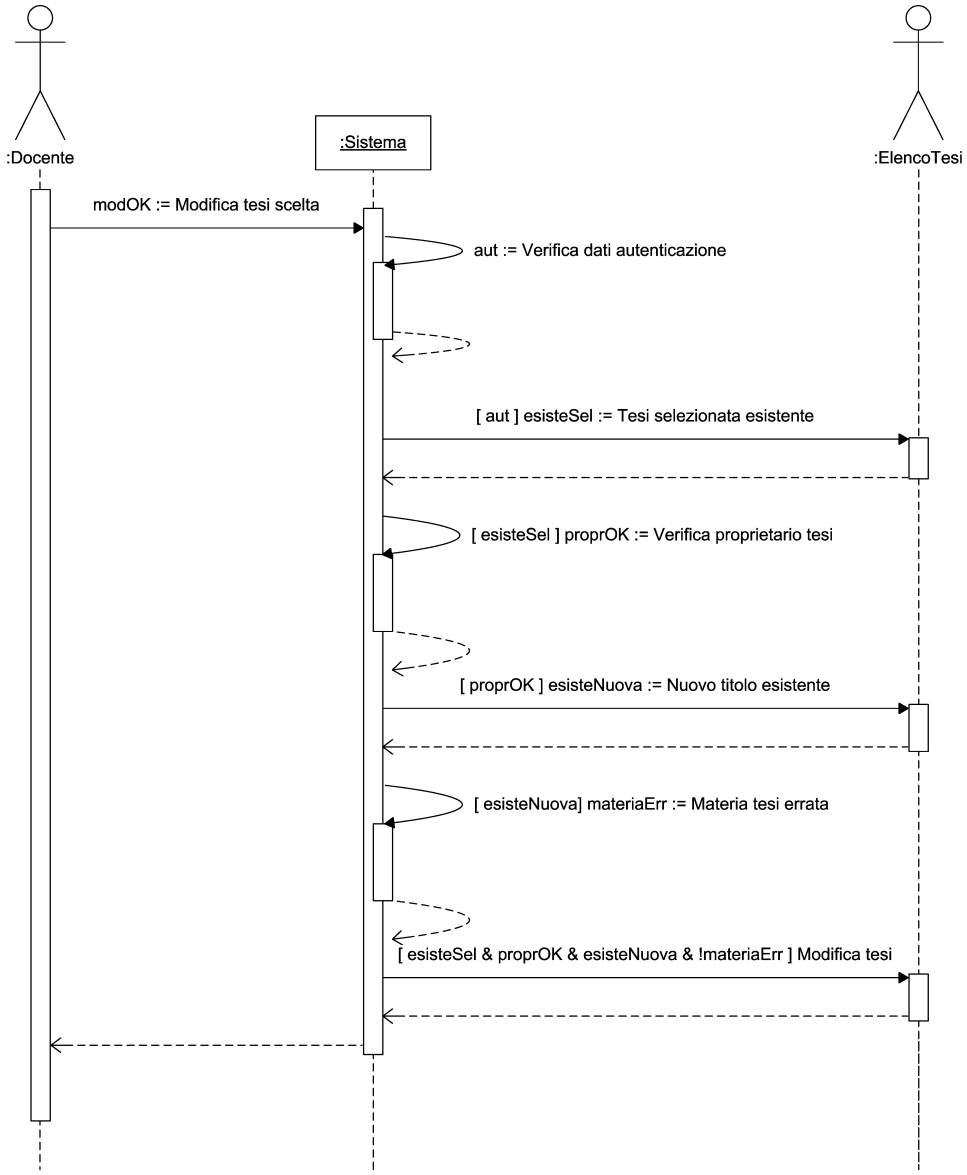


Figura 4.4: Diagramma di sequenza ad alto livello

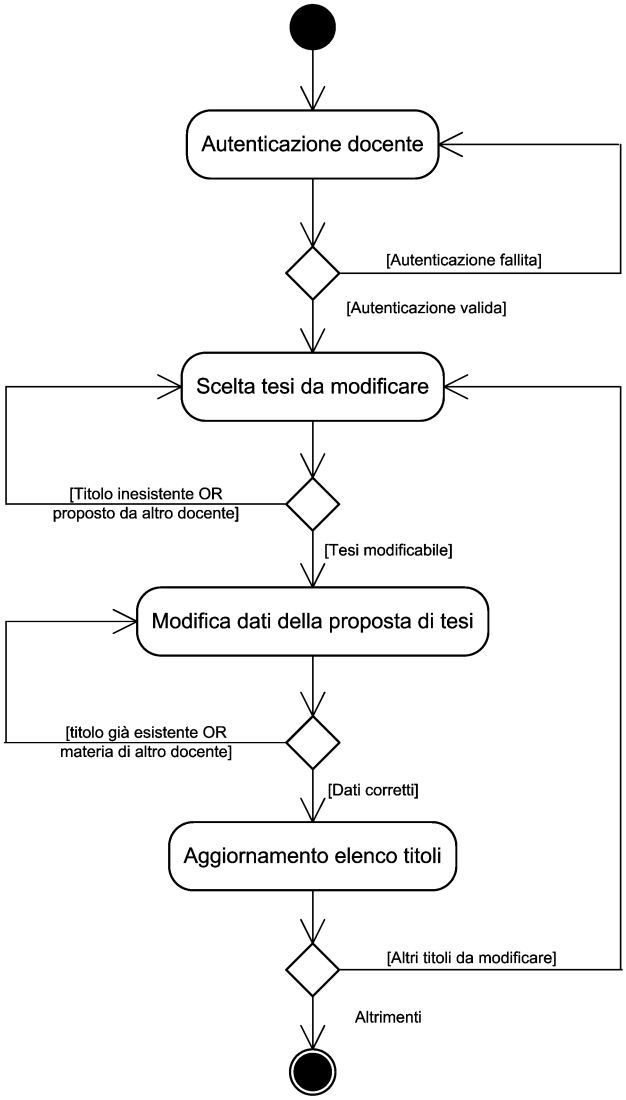


Figura 4.5: Diagramma di attività

4.3 Caso d'uso 3: Eliminazione di una proposta di tesi

Dichiarazione degli obiettivi: un docente vuole eliminare una tesi da lui proposta.

Descrizione:

a. il docente si identifica al sistema;

b. il docente indica la tesi da eliminare;

c. il sistema controlla la correttezza dei dati inseriti;

d. il sistema elimina la proposta di tesi scelta dall'elenco in cui compare.

Alternativa: *il docente non viene riconosciuto dal sistema.*

Al passo **a**, l'autenticazione al sistema ha esito negativo.

Il sistema chiede nuovamente al docente di autenticarsi.

Alternativa: *il docente chiede di eliminare un titolo non eliminabile*.

Al passo **c**, il sistema rileva che è stata scelta una tesi non eliminabile (la tesi non esiste o la proposta che si è scelto di eliminare non riguarda il docente).

Il sistema notifica l'errore permettendo di reinserire i dati correttamente.

Priorità: media

Ipotesi: Il sistema ha un elenco dei docenti, e delle tesi disponibili / assegnate.

Precondizioni: Il docente ha intenzione di eliminare una proposta di tesi da lui precedentemente pubblicata. Se la tesi da eliminare è già assegnata, il docente deve essersi accordato preventivamente con il tesista interessato.

Postcondizioni: la proposta di tesi è stata eliminata correttamente.

Problemi aperti: –

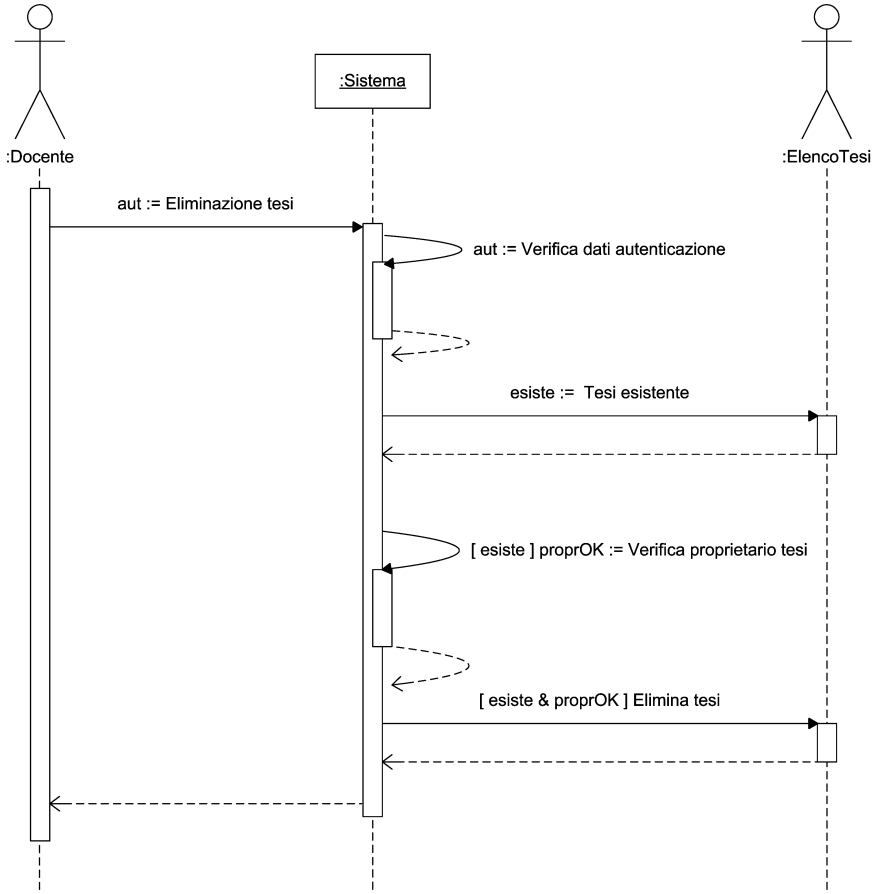


Figura 4.6: Diagramma di sequenza ad alto livello

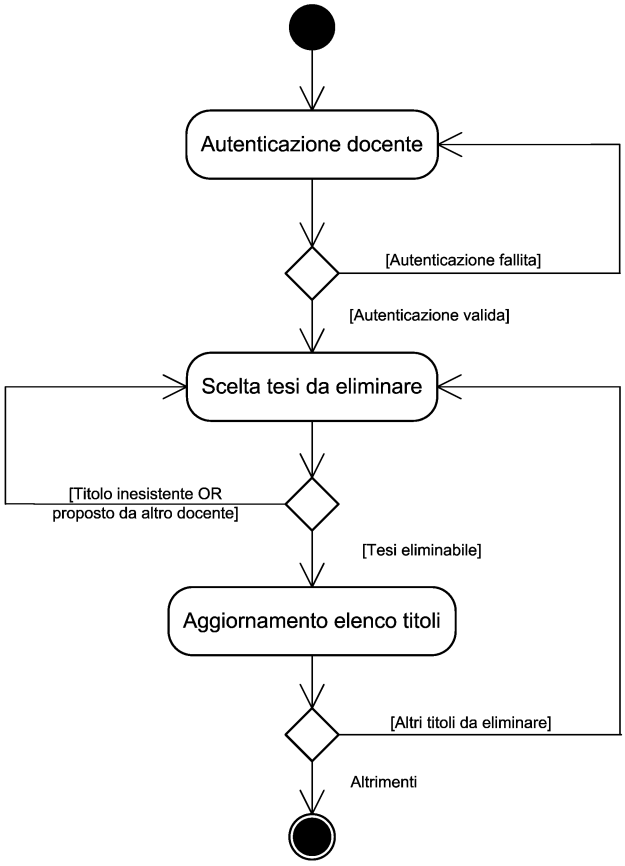


Figura 4.7: Diagramma di attività

4.4 Caso d’uso 4: Scelta della proposta di tesi

Dichiarazione degli obiettivi: uno studente, giunto al termine dei propri studi, sceglie una tesi tra quelle disponibili.

Descrizione:

a. lo studente consulta il sito per scegliere una tra le tesi disponibili;

b. nel momento in cui sceglie una tesi, lo studente deve autenticarsi al sistema;

c. Il sistema controlla che lo studente abbia i requisiti minimi per richiedere l’assegnazione della prova finale;

d. lo studente va a ricevimento dal docente proponente la tesi che lo interessa;

e. lo studente decide di affrontare la prova finale.

Alternativa: *lo studente non viene riconosciuto dal sistema*.

Al passo **b**, la procedura di autenticazione non va a buon fine.

Il sistema chiede allo studente di autenticarsi nuovamente.

Alternativa: *lo studente non è idoneo alla richiesta di assegnamento della tesi.*

Al passo **c**, il sistema rileva che lo studente non possiede i requisiti minimi per fare domanda di prova finale (non ha i 150 crediti richiesti) oppure è già assegnatario di un’altra proposta di prova finale.

Il sistema notifica l’errore.

Alternativa: *lo studente non sceglie la tesi proposta*.

Al passo **e**, a seguito di un colloquio col docente, lo studente decide che la proposta scelta non gli interessa.

Lo studente cerca un’altra proposta.

Priorità: media

Ipotesi: Il sistema ha un elenco delle tesi disponibili, e degli studenti iscritti.

Precondizioni: Lo studente ha intenzione di sostenere la prova finale.

Postcondizioni: Lo studente ha scelto una prova finale e il sistema ha inviato una Email informativa al docente che l'ha proposta.

Problemi aperti : –

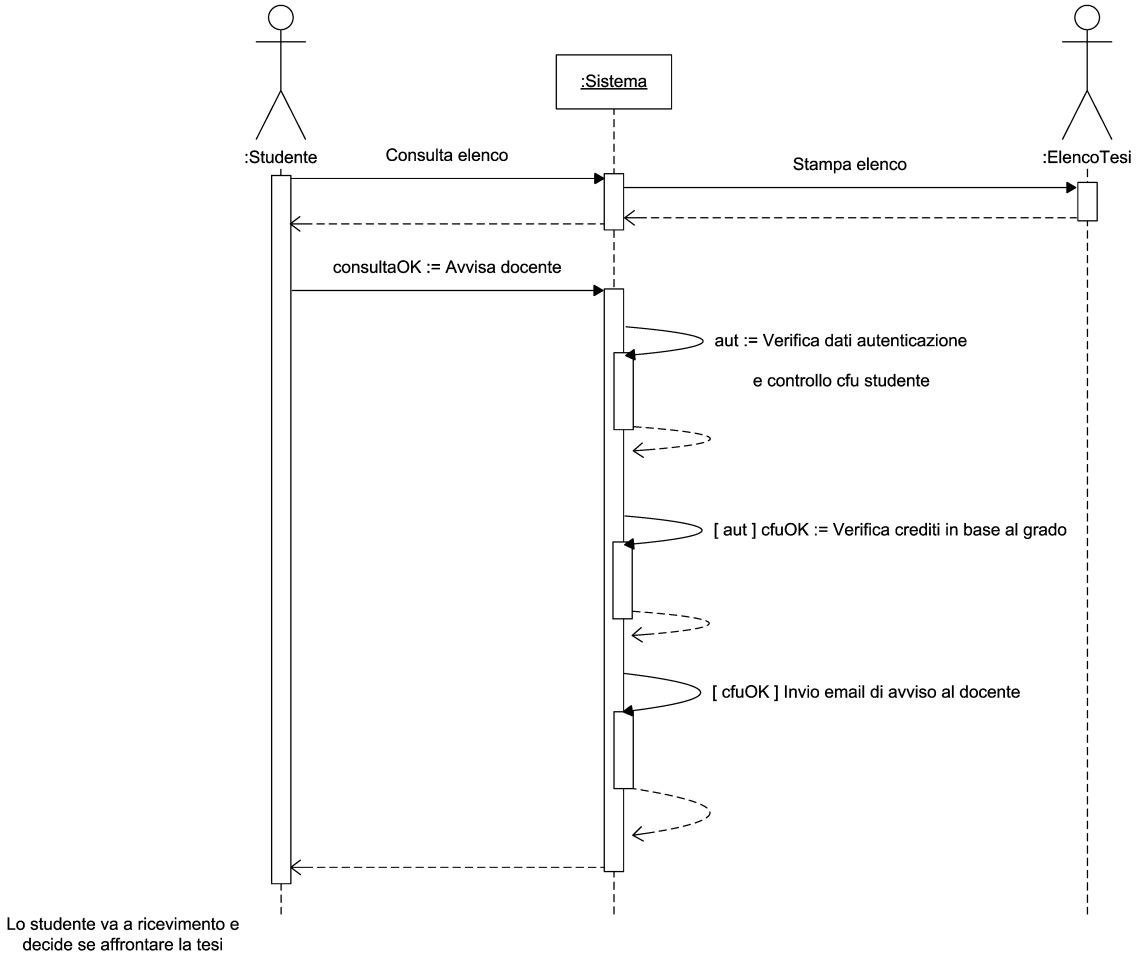


Figura 4.8: Diagramma di sequenza ad alto livello

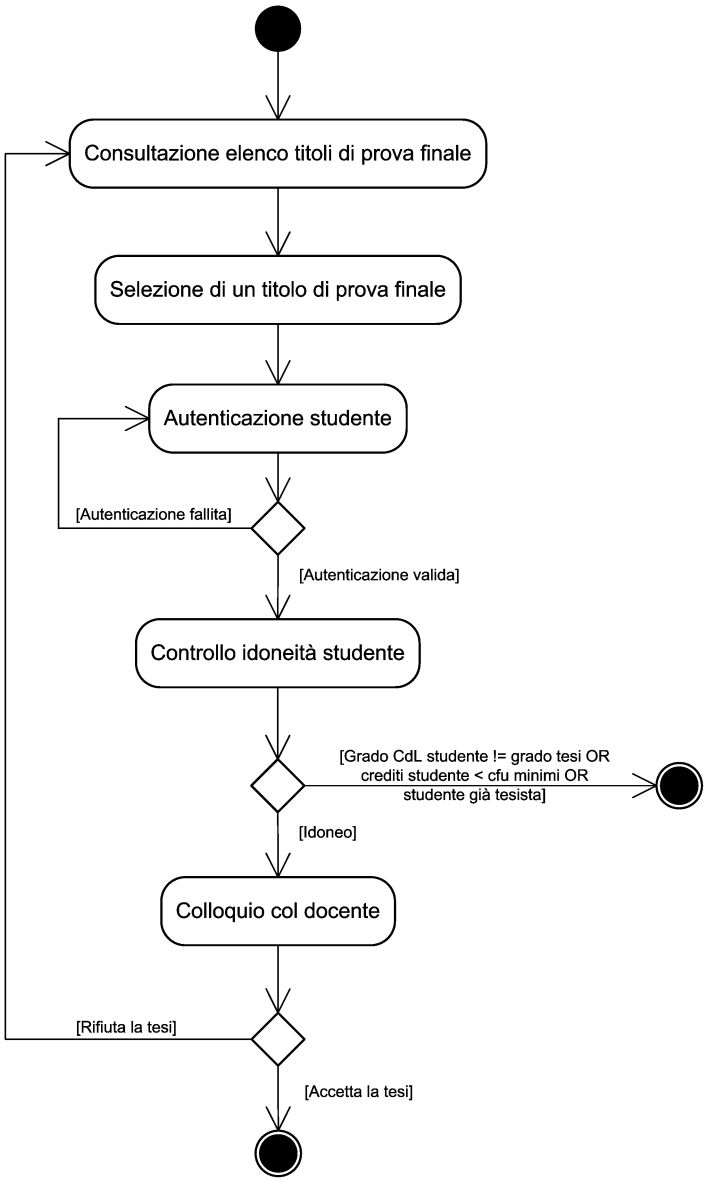


Figura 4.9: Diagramma di attività

4.5 Caso d'uso 5: Assegnazione di una proposta di tesi

Dichiarazione degli obiettivi: un docente, a seguito di un accordo con lo studente gli assegna la tesi dichiarandola non più disponibile.

Descrizione:

a. il docente si autentica al sistema;

b. il docente conferma l'assegnazione associando la proposta di tesi richiesta allo studente

c. Il sistema controlla i dati inseriti;

d. Il sistema dichiara la tesi non più come disponibile ma assegnata.

Alternativa: *il docente non viene riconosciuto dal sistema*.

Al passo **a**, l'autenticazione al sistema ha esito negativo.

Il sistema chiede nuovamente al docente di autenticarsi.

Alternativa: *i dati inseriti risultano scorretti e/o incompleti.*

Al passo **c**, il sistema rileva che si sta tentando di effettuare un'assegnazione non corretta o non consentita (la tesi e/o lo studente non esistono, si sta cercando di assegnare una tesi non più disponibile o relativa ad una materia non di competenza del docente, lo studente a cui si vuole assegnare la proposta di tesi non è in possesso dei 150 cfu o è già assegnatario di un'altra proposta di tesi).

Il sistema notifica l'errore e chiede di reinserire i dati (studente e/o prova finale) corretti.

Priorità: alta

Ipotesi: Il sistema ha un elenco delle tesi disponibili, e degli studenti iscritti.

Precondizioni: Lo studente ha fatto il colloquio con il docente e ha deciso di affrontare la tesi scelta.

Postcondizioni: la tesi è stata assegnata allo studente interessato.

Problemi aperti: –

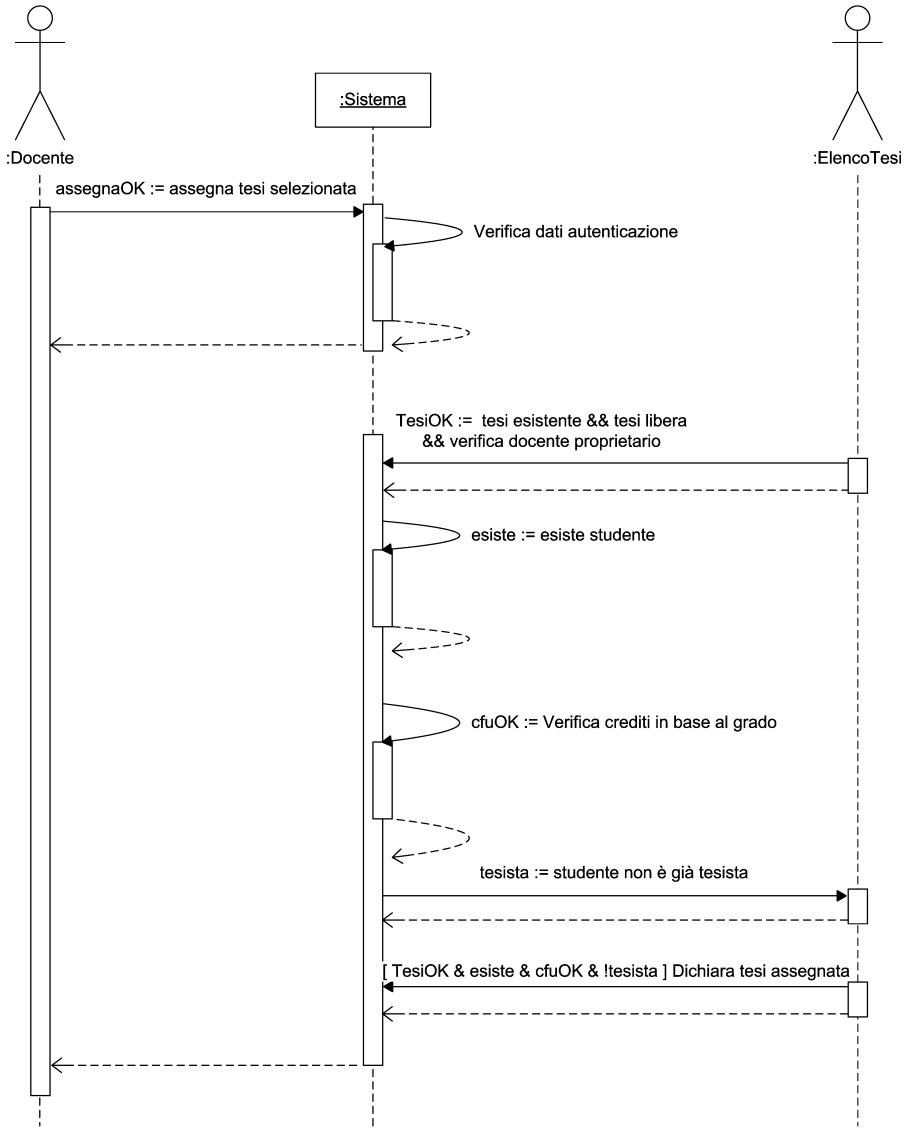


Figura 4.10: Diagramma di sequenza ad alto livello

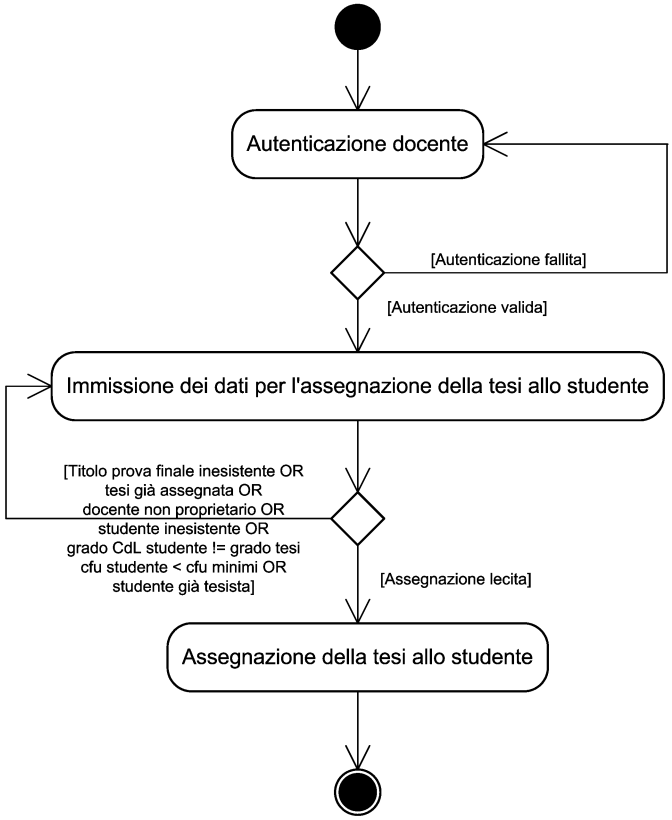


Figura 4.11: Diagramma di attività

4.6 Caso d'uso 6: Consegna della tesi in segreteria e archiviazione della stessa

Dichiarazione degli obiettivi: lo studente consegna obbligatoriamente una copia cartacea ed una elettronica della tesi in segreteria. Dopo aver verificato la coerenza tra i dati dell'elaborato consegnato (*titolo, relatore, autore, tipo di laurea, grado della tesi e tutte le altre eventuali*) con le informazioni presenti nella lista delle tesi assegnate, la copia elettronica viene resa disponibile nell'archivio informatico e la proposta di tesi viene spostata dall'elenco delle tesi assegnate a quello delle tesi svolte.

Descrizione:

a. lo studente si presenta in segreteria;

b. lo studente fornisce una copia stampata ed una copia elettronica della tesi all'impiegato;

c. l'impiegato si autentica al sistema;

d. il sistema controlla la coerenza di alcuni metadati (titolo, relatore, tesista) rispetto all'equivalente proposta di tesi presente nell'elenco delle tesi assegnate;

e. il sistema dichiara la tesi come completata (svolta) ed elimina la corrispondente proposta dall'elenco delle tesi assegnate.

Alternativa: *l'impiegato non viene riconosciuto dal sistema.*

Al passo **a**, l'autenticazione al sistema ha esito negativo.

Il sistema chiede nuovamente all'impiegato di autenticarsi.

Alternativa: *la tesi consegnata è incoerente.*

Al passo **c**, il sistema rileva che i dati nella lista di tesi assegnate non corrispondono a quelle della tesi consegnata.

Il sistema segnala un errore impedendo l'archiviazione della tesi.

Alternativa: *la consegna della tesi è successiva alla data di consegna prevista per la sessione in corso.*

Al passo **c**, il sistema controlla la sessione di laurea associata alla tesi e scopre che la data corrente (di consegna) è successiva all'ultima data utile per la consegna.

Il sistema segnala un errore impedendo l'archiviazione della tesi.

Priorità: alta

Ipotesi: il sistema ha un elenco di tesi assegnate, un elenco delle date utili per la consegna dell'elaborato di tesi per ogni sessione e un archivio delle tesi concluse.

Precondizioni: lo studente ha concluso la tesi e vuole depositarla per poter sostenere la prova finale entro i limiti di tempo previsti. Se durante lo svolgimento della tesi il titolo della stessa fosse stato modificato dallo studente, tale modifica sarebbe stata comunicata al relatore, il quale è in grado di aggiornare l'elenco delle tesi assegnate (Vedi Caso d'uso 2).

Postcondizioni: la tesi è stata rimossa dall'elenco delle tesi assegnate e inserita in quello delle tesi già svolte. La copia elettronica della tesi è stata inserita correttamente nell'apposito archivio.

Problemi aperti: Gestione dei metadati?

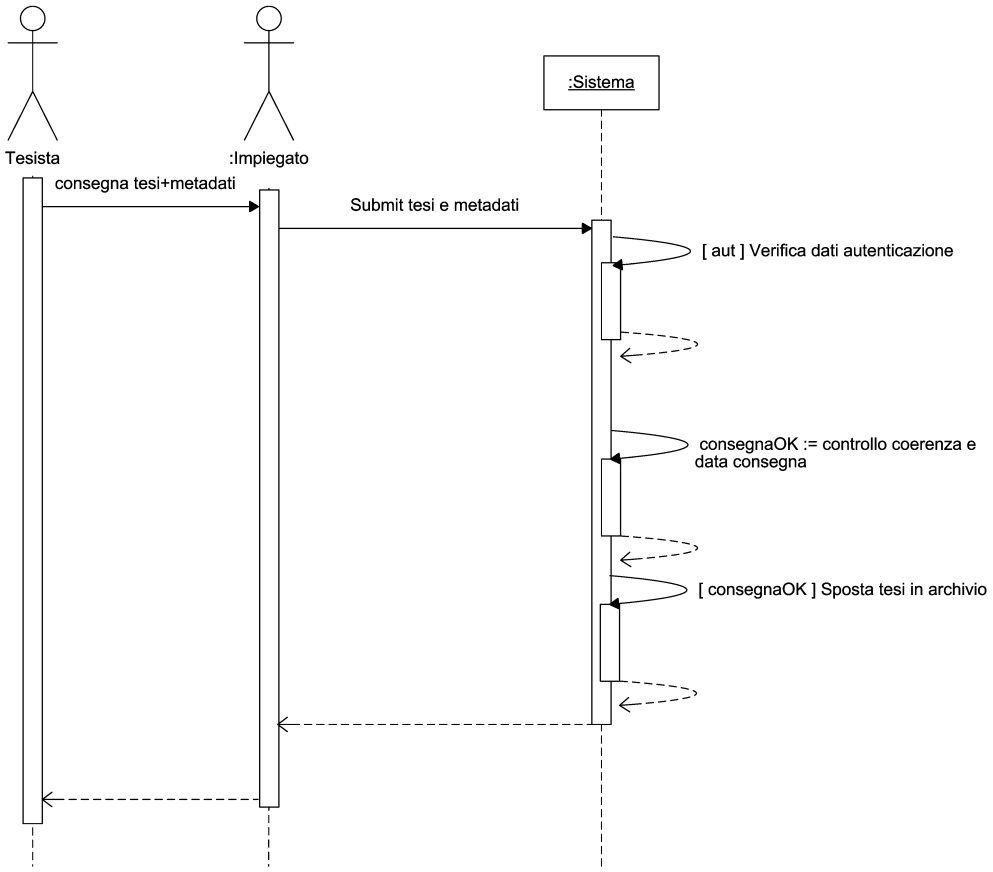


Figura 4.12: Diagramma di sequenza ad alto livello

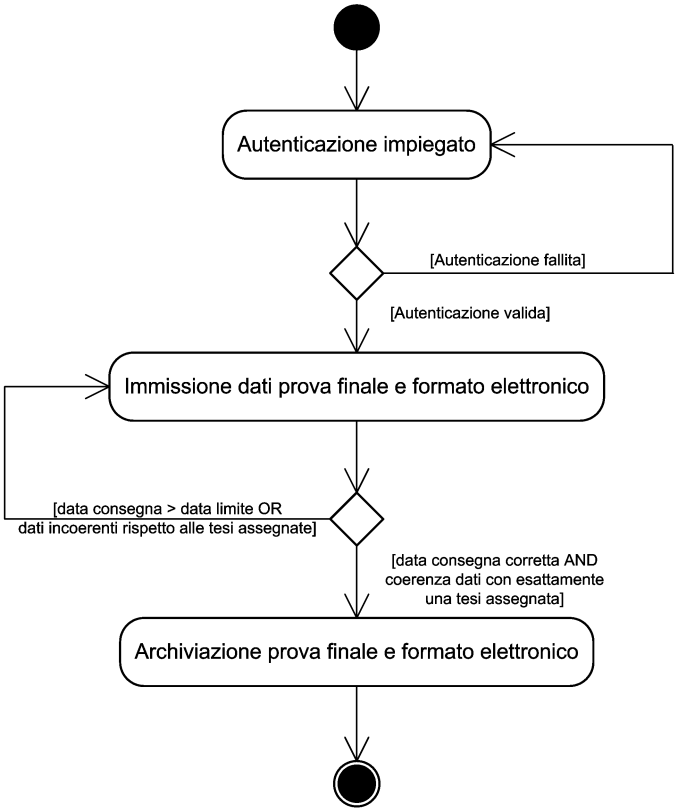


Figura 4.13: Diagramma di attività

4.7 Caso d’uso 7: Consultazione dell’archivio delle tesi

Dichiarazione degli obiettivi: Un utente autorizzato desidera consultare la versione elettronica di un eleborato di tesi.

Descrizione:

a. l’utente si identifica al sistema;

b. l’utente consulta l’archivio delle tesi concluse.

Alternativa: l’ utente non viene riconosciuto dal sistema.

Al passo **a**, l’autenticazione al sistema ha esito negativo.

Il sistema chiede nuovamente all’utente di autenticarsi.

Priorità: bassa

Ipotesi: Il sistema ha un archivio delle tesi già svolte e una serie di utenti abilitati alla consultazione.

Precondizioni: un utente vuole consultare l’archivio delle tesi già svolte.

Postcondizioni: l’utente interessato ha ottenuto una copia in formato elettronico della tesi richiesta

Problemi aperti: –

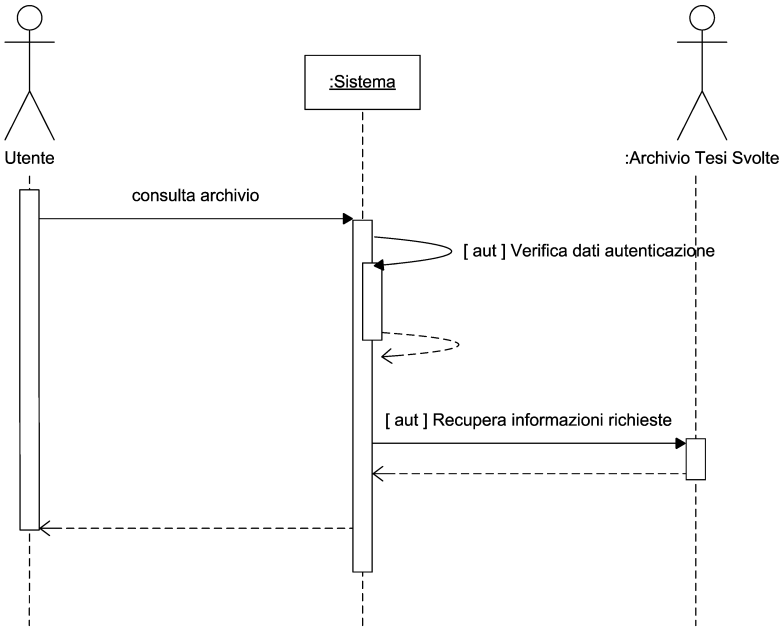


Figura 4.14: Diagramma di sequenza ad alto livello

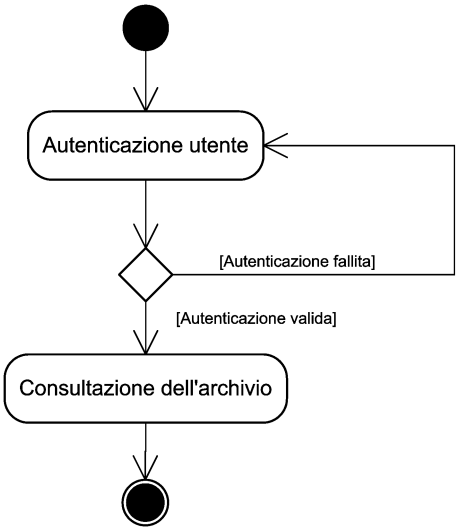


Figura 4.15: Diagramma di attività

Capitolo 5

Individuazione delle classi di analisi ad alto livello

Seguendo il metodo proposto dal libro di testo, abbiamo operato un'analisi grammaticale del documento di analisi dei requisiti (cfr. Glossario dei Termini, Cap. 3), mettendo in evidenza tutti i nomi e le frasi nominali. Tali nomi sono stati inclusi in una lista di classi potenziali. A questo punto, per ogni elemento individuato, si è verificato il soddisfacimento o meno dei sei criteri individuati da Coad e Yourdon per decidere se dovesse o meno apparire tra le classi di analisi [INGSW, pagg. 574-578]:

1. **Informazione trattenuta:** un oggetto potenziale è utile per l'analisi solo se le informazioni a esso relative devono essere memorizzate per il funzionamento del sistema;

2. **Servizi necessari:** l'oggetto potenziale deve essere dotato di operazioni che modificano in qualche modo i suoi attributi;

3. **Attributi multipli:** durante l'analisi dei requisiti, l'attenzione deve essere volta alle "informazioni primarie" (un oggetto con un solo attributo può essere utile nella progettazione, ma probabilmente è meglio rappresentato in forma di attributo di un altro oggetto durante la fase di analisi);

4. **Attributi comuni:** è possibile definire un insieme di attributi per l'oggetto potenziale che si applicano a tutte le occorrenze dell'oggetto;

5. **Operazioni comuni:** è possibile definire un insieme di operazioni per l'oggetto potenziale che si applicano a tutte le occorrenze dell'oggetto;

6. **Requisiti essenziali:** le entità esterne che compaiono nello spazio del problema e producono o consumano informazioni essenziali per le operazioni di qualsiasi soluzione sono quasi sempre definite come oggetti nel modello dei requisiti.

OGGETTO/CLASSE POTENZIALE	CLASSIFICAZIONE GENERALE
Tesi	Cosa
Crediti	Cosa
Titolo di tesi	Cosa
Studente	Entità esterna
Docente	Entità esterna
Materia	Cosa
Sessione	Cosa
Metadati	Cosa
Impiegato	Entità esterna
Commissione	Entità esterna

OGGETTO/CLASSE POTENZIALE	CRITERI RILEVANTI
Tesi	Accettato: valgono tutti i criteri
Crediti	Scartato: 3 non vale
Titolo di tesi	Scartato: 3 non vale
Studente	Accettato: valgono tutti i criteri
Docente	Accettato: valgono tutti i criteri
Materia	Scartato: 3 non vale
Sessione	Scartato: 3 non vale
Metadati	Scartato: non valgono 5 e 6
Impiegato	Accettato: valgono tutti i criteri
Commissione	Scartato: non vale nessun criterio

Riguardo all'elenco di classi individuate dalla tabella sopra riportata, è opportuno sottolineare come sostenuto anche dal libro di testo, che:

1. L'elenco non è completo, per completare il modello si dovranno aggiungere altri oggetti;
2. alcuni degli oggetti potenziali scartati diventeranno attributi di altri oggetti;
3. un enunciato diverso del problema potrebbe portare a decisioni diverse.

Nel dettaglio, abbiamo trovato infatti che alcune classi potenziali scartate sono divenute attributi di altre classi:

- **Crediti**, si è rivelato essere un attributo della classe Studente;
- **Materia**, si è rivelato essere un attributo delle classi Docente e Tesi;
- **Titolo di tesi**, si è rivelato essere un attributo della classe Tesi;
- **Sessione**, si è rivelato essere un attributo della classe Tesi;
- **Metadati**. Sono stati scartati in forza del fatto che non rappresentano altro che una sorta di raggruppamento di informazioni riguardanti la classe Tesi, non fornendo di per se' particolari funzionalità proprie, ma spostando piuttosto funzionalità e informazioni proprie della classe tesi. Per questi motivi abbiamo deciso di non creare questa ulteriore classe in fase di analisi, rimandando la decisione di una sua eventuale introduzione alla successiva fase di progettazione.

Inoltre altre classi potenziali sono state scartate proprio in forza del concetto espresso al punto 3. Difatti:

- **Commissione**, si è rivelato essere del tutto estraneo al sistema dal punto di vista delle funzionalità ad esso richieste;

Dalle considerazioni fatte finora, abbiamo ottenuto una prima bozza del diagramma delle classi di analisi ad alto livello, il quale mette in evidenza le interazioni principali e le informazioni mantenute dalle varie classi di analisi. Riportiamo qui sotto tale diagramma, del quale forniremo successivamente una discussione per motivare le scelte fatte.

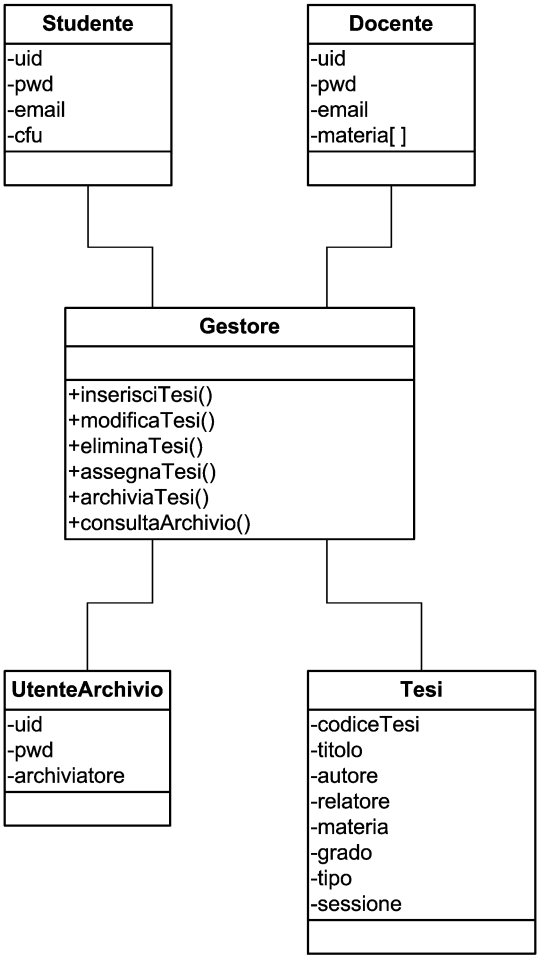


Figura 5.1: Scheletro di diagramma delle classi di analisi

Innanzitutto va ribadito che lo schema è solamente uno “schema scheletro” che verrà raffinato ed espanso nei passi successivi delle attività di analisi e di

progettazione. E' stata introdotta una classe "Gestore", non esplicitamente individuata tramite l'analisi grammaticale dei requisiti, alla quale sono state assegnate le responsabilità relative alle funzioni richieste al sistema (cfr. Casi d'Uso individuati).

Partendo da questo insieme di classi base, abbiamo iniziato un'operazione di raffinamento sia delle operazioni e degli attributi, che delle relazioni esistenti. Tale operazione ci ha portato, per passi successivi di espansione e integrazione di ogni sua parte, al diagramma delle classi finale che riporteremo più avanti, nel capitolo dedicato alla fase di progettazione.

I ruoli dell'impiegato e dell'utente che accede all'archivio, come si può ben vedere, sono stati accorpati dato che hanno le stesse funzionalità e attributi (tecnica di collassamento). Si distinguono infatti per il solo fatto di avere differenti permessi di accesso all'archivio: un utente generico può solamente accedervi per la consultazione, mentre un impiegato può effettuare inserimenti di nuove tesi (procedura di archiviazione). Per poter fare il collassamento si è rivelato necessario aggiungere un attributo booleano che indicasse, per ogni istanza di utente dell'archivio, se si trattasse di un archiviatore o meno: ci siamo serviti dell'attributo "isArchiviatore". Questa scelta ha inoltre portato benefici per quanto riguarda la leggibilità dei diagrammi, caratteristica fondamentale che non deve essere mai trascurata.

Discorso a parte merita l'analisi approfondita della classe Tesi. Infatti come si può evincere chiaramente dalle specifiche in linguaggio naturale, si possono distinguere tre tipi di tesi:

- Tesi libere (non assegnate);
- Tesi assegnate (in corso di svolgimento);
- Tesi archiviate (consegnate in segreteria);

Ad una prima approssimazione abbiamo quindi individuato l'albero di generalizzazione qui sotto riportato.

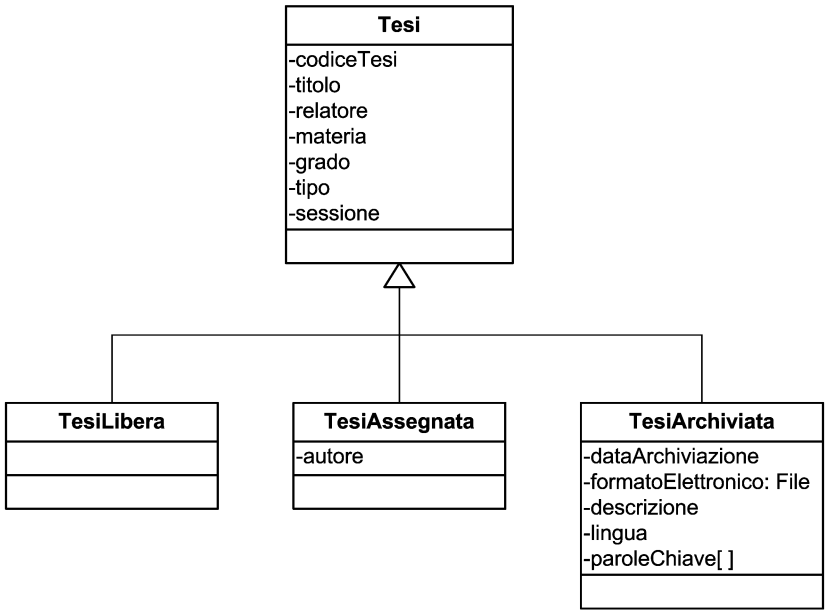


Figura 5.2: Albero di generalizzazione delle Tesi

Dato che le classi TesiLibera e TesiAssegnata differiscono per il solo attributo "tesista", abbiamo deciso di collassarle in un'unica classe, denominata TesiNonArchiviata, contenente tutti gli attributi di TesiAssegnata, dove "autore" può non essere definito (valore di default = null), e dove si è inserito un attributo booleano (il flag "assegnata"). Si è scelto invece di tenere separata la classe TesiArchiviata in quanto sostanzialmente differente dalle due appena descritte (in particolare si noti la presenza degli attributi: formatoElettronico, una sorta di indicatore univoco di risorsa su disco contenente la tesi in formato elettronico, e delle parolechiave, utili per eseguire eventuali ricerche in archivio).

L’albero d’ereditarietà che si è ricavato dalle osservazioni appena fatte è il seguente:

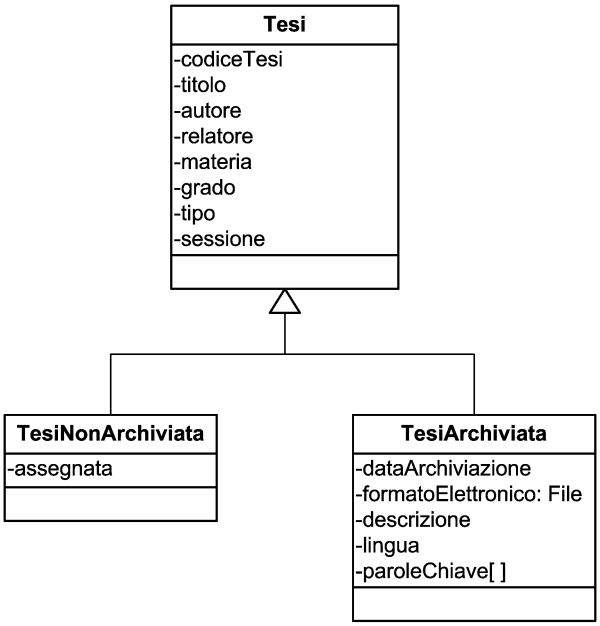


Figura 5.3: Albero di generalizzazione delle tesi raffinato

Ed ecco il diagramma delle classi finale che abbiamo ottenuto a seguito di tutte le considerazioni fin qui fatte:

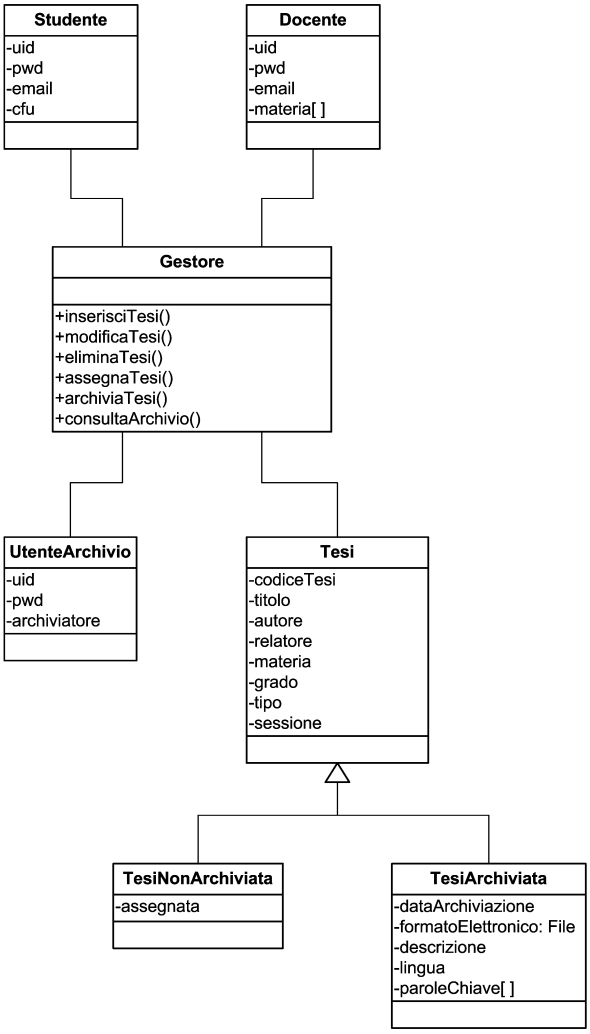


Figura 5.4: Diagramma delle Classi Analisi completo

Capitolo 6

Scelta dell’architettura software

Per realizzare l’applicazione abbiamo scelto un modello di architettura a livelli, in modo da poterne sfruttare i vantaggi intrinseci di modularità, protezione e incapsulamento della complessità. Come si può notare le relazioni di dipendenza tra i package dei vari livelli sono tutte nella stessa direzione (dal package al livello più alto verso quello a livello più basso), caratteristica tipica delle architetture software a livelli. E’ stata posta una particolare cura nell’evitare di usare un numero troppo elevato di livelli, cosa che avrebbe potuto causare penalizzazioni in termini prestazionali. Al livello più alto, e quindi in diretta comunicazione con gli utenti esterni al sistema, troviamo il package di gestione e controllo, che si occupa in primo luogo di fornire le funzionalità richieste al sistema ed individuate con l’analisi dei casi d’uso. Inoltre esso implementa le politiche di controllo degli accessi e di verifica della correttezza dei dati: praticamente si mette in atto la "logica di business" e gestisce in maniera controllata l’accesso ai dati. Al livello sottostante troviamo la rappresentazione di tutte le strutture dati necessarie al funzionamento del sistema e le classi container per gli elenchi. Sono i "mattoncini elementari" che, manipolati dalle classi di livello superiore, fanno da repository di informazioni. Ad un livello ancora sottostante possiamo immaginare di trovare un package di utility, utile in fase di implementazione, contenente classi di sistema per l’interfaccia grafica, per l’invio di Email tramite server POP3, e per la gestione della persistenza delle informazioni rappresentate al livello superiore (i.e. classi di gestione di database).

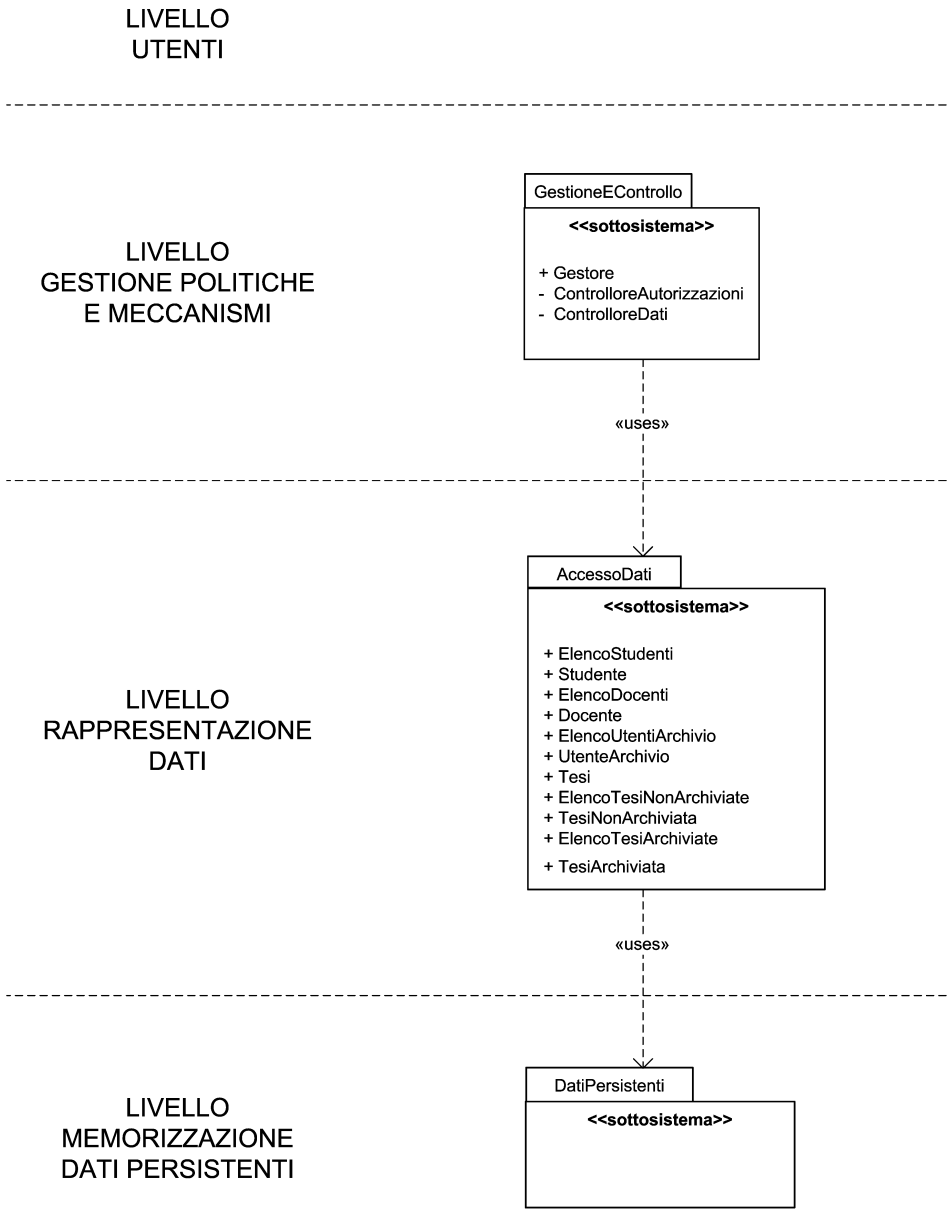


Figura 6.1: Diagramma dei Package

Capitolo 7

Package di Analisi

Come si può notare dal diagramma sovrastante, i package sono stati individuati cercando di mantenere più alto possibile il grado di coesione e abbassare quello di accoppiamento. Come si può notare infatti, non ci sono relazioni che introducono un alto grado di interdipendenza tra classi di package diversi. Si ricorda infatti che [UML pagg. 180-181] "le classi sono correlate nel modo più forte dall'ereditarietà, quindi dalla composizione, dall'aggregazione e, infine, dalle dipendenze." e che "Le classi appartenenti a gerarchie di ereditarietà o di composizione sono le migliori candidate per essere raggruppate in uno stesso package". Tenendo a mente queste linee guida per l'individuazione dei package di analisi siamo giunti alla soluzione proposta, che ci ha portato ad avere soltanto relazioni di dipendenza tra classi di package diversi, riducendo quindi al minimo l'accoppiamento del sistema. Tutto questo, nel rispetto della regola pratica che consiglia di non superare le 10 classi per package, ci ha portato ad una scomposizione dei package come moduli funzionali distinti e fortemente coesi. Infine grazie alle caratteristiche intrinseche dell'architettura software scelta siamo riusciti ad evitare le dipendenze circolari tra package in maniera del tutto automatica.

Parte III

Progettazione

Capitolo 8

Design Pattern

Prima di procedere nel raffinamento del diagramma delle classi di analisi, abbiamo cercato di individuare eventuali design patterns applicabili al nostro progetto, in modo tale da risparmiare tempo e avere delle buone soluzioni già pronte grazie al riutilizzo di conoscenze maturate nel corso degli anni dalla comunità dei progettisti software. Come punto di riferimento dal quale scegliere eventuali design patterns riutilizzabili, abbiamo preso il "Portland Pattern Repository" [PPR01], consigliato anche dal nostro libro di testo.

Tra i vari schemi progettuali presentati nel sito web della Portland, ne abbiamo individuati due particolarmente utili. Proponiamo nella tabella sottostante una breve descrizione di ogni design pattern individuato e dei punti in cui è possibile applicarlo.

Design Pattern	Descrizione	Applicazione
Singleton	Quando un sistema ha bisogno soltanto di una sola istanza di una classe, e quella istanza deve essere accessibile da varie parti del sistema, allora si può controllare sia l'istanziazione che l'accesso alla classe stessa definendola come Singleton.	Tutte le classi del package GestioneEControllo: Gestore, ControlloreDati e ControlloreAutorizzazioni
Iterator	Fornisce un oggetto che permette di "visitare" una qualche struttura aggregata, astraendo dalle assunzioni riguardanti l'implementazione della struttura stessa. Il più semplice tipo di iteratore ha un metodo "next element", il quale restituisce gli elementi in un ordine sequenziale finchè non ce ne sono più. Iteratori più sofisticati possono permettere movimenti di tipo differente e molteplici direzioni di visita della struttura complessa.	Tutte le classi container: elenco Docenti, Studenti, Tesi ...

Capitolo 9

Descrizione delle classi di progettazione

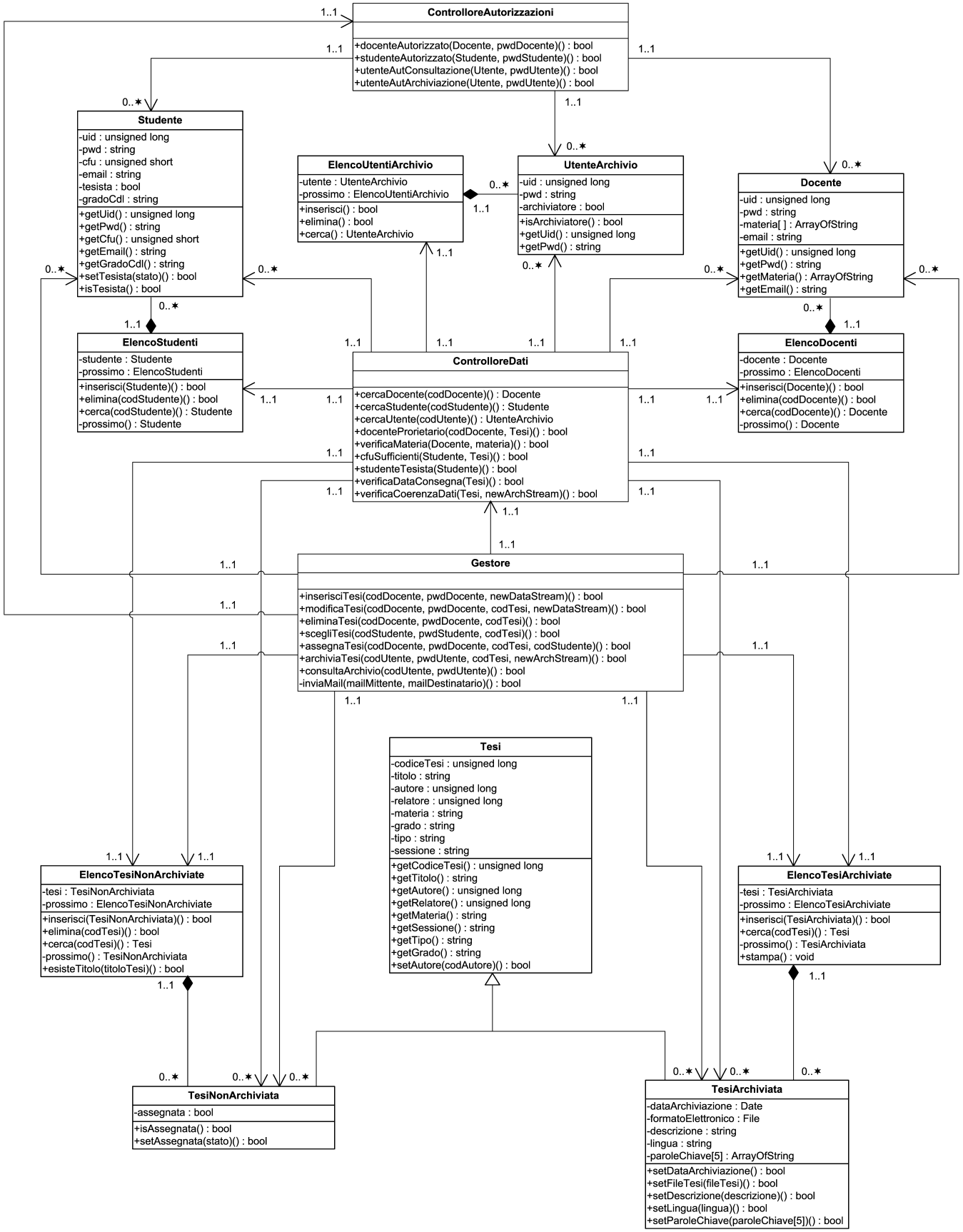


Figura 9.1: Diagramma delle classi

9.1 CLASSI DI GESTIONE E CONTROLLO

9.1.1 Gestore

Descrizione: questa classe ha lo scopo di mettere a disposizione dell'utente finale le funzionalità principali fornite dall'applicazione, individuate con l'analisi dei casi d'uso. Per portare a termine tali compiti si appoggia a classi che effettuano i controlli, in modo da verificare la leicità dell'operazione invocata.

Attributi:

Operazioni:

- inserisciTesi() prende come parametri un codice docente e la password corrispondente, in modo da controllare che l'utente che sta chiedendo di inserire una nuova proposta di prova finale sia effettivamente autorizzato a farlo. Il terzo ed ultimo parametro è uno stream di dati, ricevuti in input tramite form, contenenti le informazioni relative alla nuova tesi da inserire. Queste informazioni verranno utilizzate per inizializzare la nuova tesi creata.

- modificaTesi(): prende come primi due parametri il codice docente (codDoc) e la password, allo scopo di verificare le credenziali di autorizzazione. Il terzo parametro è un codice di tesi, che individua la tesi esistente da modificare. Infine il quarto parametro, come per il metodo precedente, è un datastream contenente le nuove informazioni che andranno a sovrascrivere quelle della tesi esistente.

- eliminaTesi(): il metodo permette, se si è autorizzati a farlo (si controlla tramite i primi due parametri codDocente e password), di eliminare la tesi individuata da codiceTesi.

- scegliTesi(): questo metodo permette ad uno studente (individuato dal primo parametro codiceStudente), di prendere visione dell'elenco di proposte

di tesi e, se autorizzato a farlo (lo si verifica tramite il secondo parametro Password), di scegliere la tesi individuata da codTesi. La scelta non comporta l'assegnazione definitiva e va confermata dal docente previo incontro con lo studente.

- assegnaTesi(): questo metodo permette ad un docente, se autorizzato a farlo (parametri codDocente e pwdDocente), di assegnare una tesi (terzo parametro codTesi) ad uno studente (quarto parametro codStudente).

- archiviaTesi(): questo metodo permette a un impiegato (le cui credenziali si verificano usando i primi due parametri codUtente e pwdUtente) di archiviare la tesi univocamente identificata da codTesi (terzo parametro). Il quarto parametro è un data stream contenente le informazioni riportate nella copia cartacea della tesi (lingua, titolo, relatore, materia etc...) che verranno confrontate dal punto di vista della coerenza con le informazioni della tesi recuparbile tramite "codTesi", presente nell'elenco delle tesi assegnate.

- consultaArchivio(): questo metodo permette ad un utente di visionare l'archivio delle tesi già consegnate in segreteria. I due parametri codUtente e pwdUtente permettono di controllare le credenziali di autorizzazione.

9.1.2 ControlloreDati

Descrizione: questa classe fornisce alla classe Gestore una serie di metodi di appoggio che consentono di verificare la correttezza dei dati forniti in input e la leicità delle operazioni invocate.

Attibuti:

Opeazioni:

- cercaDocente(): prende come unico parametro un codice di docente e ritorna l'istanza della classe individuata da quel codice unico.

- cercaStudente(): come sopra ma opera su istanze di "Studente".

- cercaUtente(): come sopra ma opera su istanze di "Utente".

- docenteProprietario(): verifica che il docente individuato da codDocente (primo parametro) sia proprietario dell'istanza di tesi passata come secondo parametro. Utile nei casi di modifica ed eliminazione di una proposta di tesi, per controllare che i dati inseriti identifichino realmente una tesi di cui il docente è proprietario.

- verificaMateria(): verifica che l'istanza di Docente passata come primo parametro sia effettivamente titolare del corso passato come secondo parametro. Utile per evitare che un docente pubblichi una tesi per una materia della quale non è titolare.

- cfuSufficienti(): verifica che l'istanza di studente passata come primo parametro abbia i cfu sufficienti per fare richiesta della prova finale individuata dall'istanza di Tesi passata come secondo parametro. Il controllo infatti permette di fare controlli sia per tesi triennali che quinquennali, in base al tipo di immatricolazione dello studente e al tipo di tesi. A seconda dei casi cambia la soglia di crediti necessari.

- studenteTesista(): restituisce un booleano che indica se l'istanza di studente passata come unico parametro sia un tesista o meno.

- verificaDataConsegna(): metodo che permette di verificare che una tesi (istanza di tesi passata come parametro) non sia stata consegnata oltre il tempo limite fissato per la sessione corrente.

- verificaCoerenzaDati(): permette di verificare, in sede di consegna in segreteria della tesi svolta, la coerenza dei dati presenti nella copia cartacea della stessa con le informazioni già presenti per quella tesi nella lista delle tesi assegnate. Prende come unico parametro l'istanza di tesi che si sta consegnando.

9.1.3 ControlloreAutorizzazioni

Descrizione: questa classe fornisce alla classe Gestore altri metodi di appoggio ma, a differenza di quelli forniti da ControlloreDati, questi sono metodi

specifici per la verifica delle credenziali di autorizzazione delle varie classi di utenti del sistema.

Attributi:

Operazioni:

- docenteAutorizzato(): verifica che l'istanza di Docente passata come primo parametro abbia fornito una password corretta (secondo parametro pwdDocente).
- studenteAutorizzato(): verifica che l'istanza di Studente passata come primo parametro abbia fornito una password corretta (secondo parametro pwdStudente).
- utenteAutConsultazione(): verifica che l'istanza di Utente passata come primo parametro abbia fornito una password corretta (secondo parametro pwdUtente), al fine di consultare l'archivio delle tesi svolte.
- utenteAutArchiviazione(): verifica che l'istanza di Utente passata come primo parametro abbia fornito una password corretta (secondo parametro pwdUtente), al fine di archiviare una nuova tesi (l'utente è un impiegato della segreteria).

9.2 CLASSI DI RAPPRESENTAZIONE DATI

Le classi "elenco", sono tutte uguali dal punto di vista strutturale, con la sola differenza che operano su dati di tipo differente (ElencoDocenti opera su Docente, ElencoStudente su Studente etc...). Per non rendere la lettura troppo pesante riportiamo qui la descrizione di una sola delle classi "elenco", le altre sono analoghe.

9.2.1 ElencoStudenti

Descrizione: questa classe rappresenta un elenco di elementi di tipo "Studente". E' una cosiddetta Classe Container.

Attributi:

- studente : istanza di studente che rappresenta l'elemento corrente dell'elenco.
- prossimo : attributo di tipo ElencoStudenti, è un puntatore al prossimo elemento dell'elenco.

Operazioni:

- inserisci(): permette di inserire un nuovo elemento di tipo Studente nell'elenco.
- elimina(): permette di eliminare lo studente individuato dal parametro codStudente.
- cerca(): restituisce, se presente nell'elenco, l'istanza di Studente individuata dal parametro codStudente.
- prossimo(): permette di scorrere l'elenco, passando all'elemento successivo. Fornisce la funzionalità di iteratore.

9.2.2 Studente

Descrizione: questa classe modella l'entità studente, e contiene tutte le informazioni ad essa relative, utili all'implementazione dei meccanismi necessari a mettere in atto le politiche di business individuate in fase di analisi dei requisiti.

Attributi:

- uid(tipo numerico): è un codice identificativo univoco dello studente (in questo caso il numero di matricola), che verrà utilizzato anche come username per accedere ai servizi offerti dall'applicazione.
- pwd(tipo stringa): password associata allo uid, necessaria per la verifica delle credenziali di autenticazione in fase di login.

- cfu(tipo numerico): numero di Crediti Formativi acquisiti dallo studente.
- email(tipo stringa): indirizzo e-mail dello studente, utile per comunicazioni di segreteria o per interagire con i docenti.
- tesista(tipo booleano): flag che indica se lo studente è assegnatario di una tesi o meno.
- gradoCdl(tipo stringa): indica il tipo di studente in base alla sua iscrizione: triennale, specialistica o vecchio ordinamento.

Operazioni:

Come per tutte le altre classi di rappresentazione di dati in memoria, sono previsti metodi getter e setter per ogni attributo.

9.2.3 Docente

Descrizione: questa classe modella l'entità docente, e contiene tutte le informazioni ad essa relative, utili all'implementazione dei meccanismi necessari a mettere in atto le politiche di business individuate in fase di analisi dei requisiti.

Attributi:

- uid(tipo numerico): è un codice identificativo univoco del docente (ad esempio si potrebbe trattare del codice docente utilizzato anche nella compilazione delle domande di prova finale), che verrà utilizzato anche come username per accedere ai servizi offerti dall'applicazione.
- pwd(tipo stringa): password associata allo uid, necessaria per la verifica delle credenziali di autenticazione in fase di login.
- materia[] (array di stringhe): contiene le informazioni relative al corso o ai corsi di cui il docente è titolare.
- email(tipo stringa): indirizzo e-mail del docente.

Operazioni:

- come per tutte le altre classi di rappresentazione di dati in memoria, sono previsti metodi getter e setter per ogni attributo.

9.2.4 UtenteArchivio

Descrizione: questa classe modella l'entità UtenteArchivio, e contiene tutte le informazioni ad essa relative, utili all'implementazione dei meccanismi necessari a mettere in atto le politiche di business individuate in fase di analisi dei requisiti.

Attributi:

- uid(tipo numerico): è un codice identificativo univoco dell'utente, che verrà utilizzato anche come username per accedere ai servizi offerti dall'applicazione.
- pwd(tipo stringa): password associata allo uid, necessaria per la verifica delle credenziali di autenticazione in fase di login.
- archiviatore (tipo booleano): flag che indica se l'utente è autorizzato ad archiviare tesi completate (trattasi di impiegato o suo delegato), o può solamente prendere visione del contenuto dell'archivio.

Operazioni:

- come per tutte le altre classi di rappresentazione di dati in memoria, sono previsti metodi getter e setter per ogni attributo.

9.2.5 Tesi

Descrizione: questa classe modella l'entità Tesi e contiene tutte le informazioni ad essa relative, utili all'implementazione dei meccanismi necessari a mettere in atto le politiche di business individuate in fase di analisi dei requisiti. Questa classe è padre delle due classi TesiArchiviata e TesiNonArchiviata, e ne fattorizza attributi e metodi comuni, per una maggior robustezza e manutenibilità del codice.

Attributi:

- codiceTesi(tipo numerico): è un codice identificativo univoco della tesi.
- titolo(tipo stringa): titolo della proposta di prova finale.
- autore(tipo numerico): riferimento al codice dello studente al quale viene assegnata la tesi. Al momento di creazione di una nuova tesi tale attributo è null.
- relatore(tipo numerico): riferimento al codice del docente proponente la prova finale.
- materia(tipo stringa): materia di interesse della prova finale. Deve essere una delle materie di cui il docente proponente è titolare.
- grado(tipo stringa): indica se è una tesi triennale, specialistica o per il vecchio ordinamento.
- tipo(tipo stringa): indica di quale tipologia di tesi si tratta: di rassegna, implementativa, di ricerca etc...
- sessione(tipo stringa): indica la sessione entro la quale si intende portare a termine e consegnare la prova finale.

Operazioni:

- come per tutte le altre classi di rappresentazione di dati in memoria, sono previsti metodi getter e setter per ogni attributo.

9.2.6 TesiNonArchiviata

Descrizione: questa classe modella l'entità Tesi, e contiene tutte le informazioni ad essa relative, utili all'implementazione dei meccanismi necessari a mettere in atto le politiche di business individuate in fase di analisi dei requisiti. Questa classe è figlia della classe Tesi, alla quale aggiunge alcuni attributi specifici delle tesi assegnate ma non ancora concluse.

Attributi (oltre a quelli ereditati da Tesi):

- assegnata(tipo booleano): flag che permette di distinguere tra tesi libera e tesi già assegnata ad uno studente.

Operazioni:

- come per tutte le altre classi di rappresentazione di dati in memoria, sono previsti metodi getter e setter per ogni attributo.

9.2.7 TesiArchiviata

Descrizione: questa classe modella l'entità Tesi, e contiene tutte le informazioni ad essa relative, utili all'implementazione dei meccanismi necessari a mettere in atto le politiche di business individuate in fase di analisi dei requisiti. Questa classe è figlia della classe Tesi, alla quale aggiunge alcuni attributi specifici delle tesi concluse.

Attributi (oltre a quelli ereditati da Tesi):

- dataArchiviazione(tipo data): indica la data in cui la tesi è stata consegnata in segreteria.
- fileTesi(tipo file): contiene un identificatore univoco di risorsa che punta alla versione elettronica della tesi.
- descrizione(tipo stringa): contiene un abstract della tesi.
- paroleChiave[5](tipo array di stringhe): contiene le cinque parole chiave associate alla tesi consegnata.

Operazioni:

- come per tutte le altre classi di rappresentazione di dati in memoria, sono previsti metodi getter e setter per ogni attributo.

Capitolo 10

Realizzazione dei casi d’uso

10.1 Il caso d’uso 1

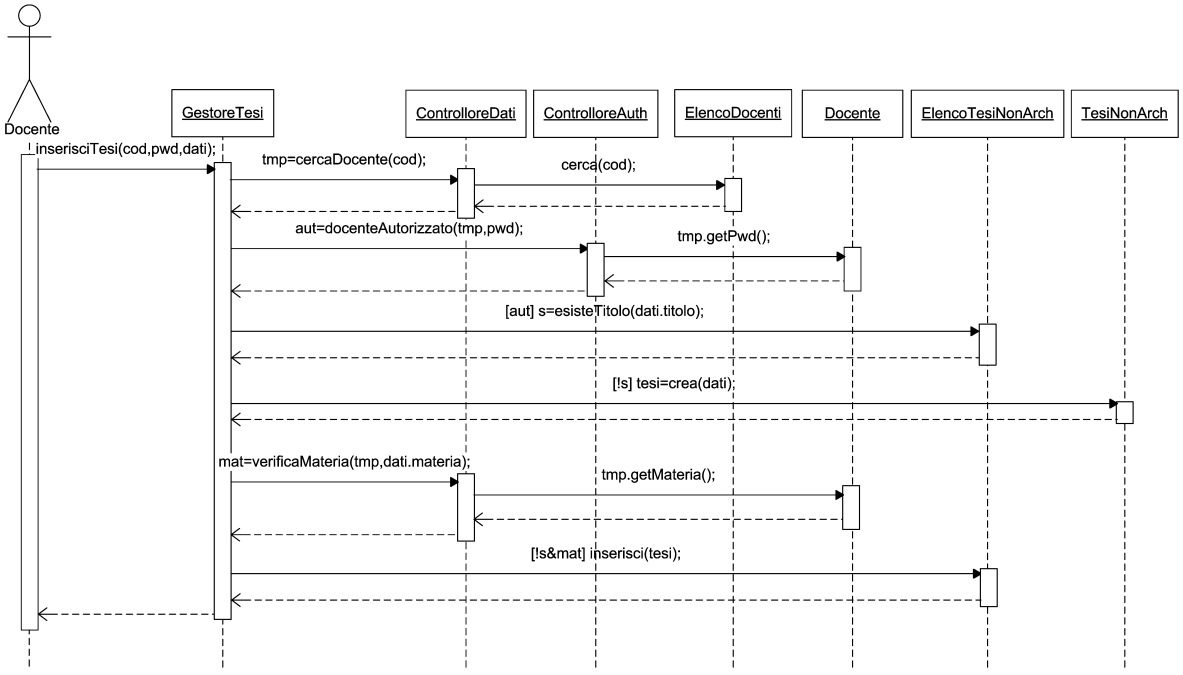


Figura 10.1: Sequence caso d’uso 1

Il docente invoca il metodo *inserisciTesi* di GESTORE, dando inizio al caso d'uso "inserimento di una nuova proposta di tesi". Tale metodo prende come parametri un codice univoco di docente (userID) e relativa password (informazioni essenziali per la successiva verifica delle credenziali di autenticazione). Come terzo parametro, viene fornito dal docente un DataStream contenente il flusso di informazioni della nuova tesi da inserire (titolo della tesi, grado della tesi, materia ecc..). A questo punto il controllo passa al GESTORE, il quale lo ritornerà al docente solo al termine delle operazioni di inserimento, ritornando un valore booleano recante indicazioni circa il buon esito o meno delle stesse. Appena ottenuto il controllo, il GESTORE si preoccuperà di ottenere l'istanza della classe DOCENTE corrispondente ai dati di login ricevuti: per farlo chiama il metodo *cercaDocente* della classe CONTROLLOREDATI, la quale farà da intermediario girando la richiesta di ricerca all'ELENCODOCENTI e ritornando l'eventuale istanza trovata al GESTORE. A questo punto il controllo ritorna al GESTORE il quale, in caso di ricerca con successo, utilizzerà l'istanza di docente appena ottenuta per demandare al CONTROLLOREAUTENTICAZIONE la verifica delle credenziali. Per fare questo, viene appunto invocato il metodo *docenteAutorizzato* di CONTROLLOREAUTENTICAZIONE, passando come parametri l'istanza di Docente appena trovata e la password inserita inizialmente dal docente. Il CONTROLLOREAUTENTICAZIONE si preoccuperà quindi di verificare la corrispondenza tra la password inserita e quella memorizzata all'interno della classe Docente (attributo pwd). Se la verifica delle credenziali avrà successo, il metodo *docenteAutorizzato* ritornerà TRUE e quindi il GESTORE continuerà ad effettuare gli opportuni controlli, dal buon esito dei quali dipenderà l'effettivo inserimento della nuova proposta di tesi. Si controlla infatti che il titolo della tesi che si vuole inserire (dato reperibile all'interno del DataStream) non esista già nell'elenco delle proposte di tesi: infatti non possono esserci due tesi in corso di svolgimento con lo stesso titolo. Questa funzionalità viene ottenuta chiamando il metodo *esisteTitolo* della classe ELENCOTESINONARCHIVIATE, alla quale viene passato come parametro il titolo della tesi estrapolato dal DataStream. Se il titolo non esiste già nell'elenco, il GESTORE procede a creare una nuova istanza di TESINONARCHIVIATA, chiamando il costruttore della classe stessa passandogli come parametri le informazioni di inizializzazione contenute nel DataStream. Una volta creata la nuova

istanza di TESINONARCHIVIATA, si verifica che il docente abbia inserito una materia di sua competenza come materia della nuova proposta di tesi: infatti si vuole impedire l'inserimento di una tesi la cui materia non sia tra quelle di cui il docente è titolare. Per fare questo, il GESTORE invoca il metodo *verificaMateria* della classe CONTROLLOREDATI, passando come parametri: l'istanza di docente che ha già ottenuto in precedenza, e la materia estratta dal DataStream. Il controllo passa quindi al CONTROLLOREDATI, il quale non fa altro che controllare la presenza della materia fornita nel DataStream all'interno dell'array di materie di cui il docente è titolare (attributo materia[] della classe DOCENTE). Se anche questo controllo ha esito positivo, il GESTORE procede all'effettivo inserimento chiamando il metodo *Inserisci* della classe ELENCOTESINONARCHIVIATE. Tale metodo ha come parametro la sola istanza di *Tesi*, già creata in precedenza dal GESTORE.

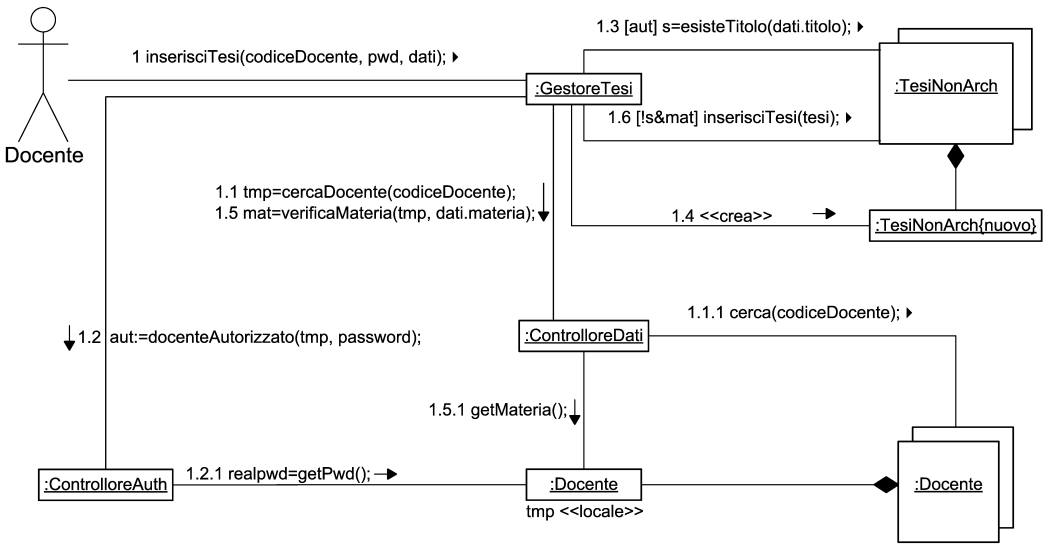


Figura 10.2: Diagramma di collaborazione del caso d'uso 1

10.2 Il caso d'uso 2

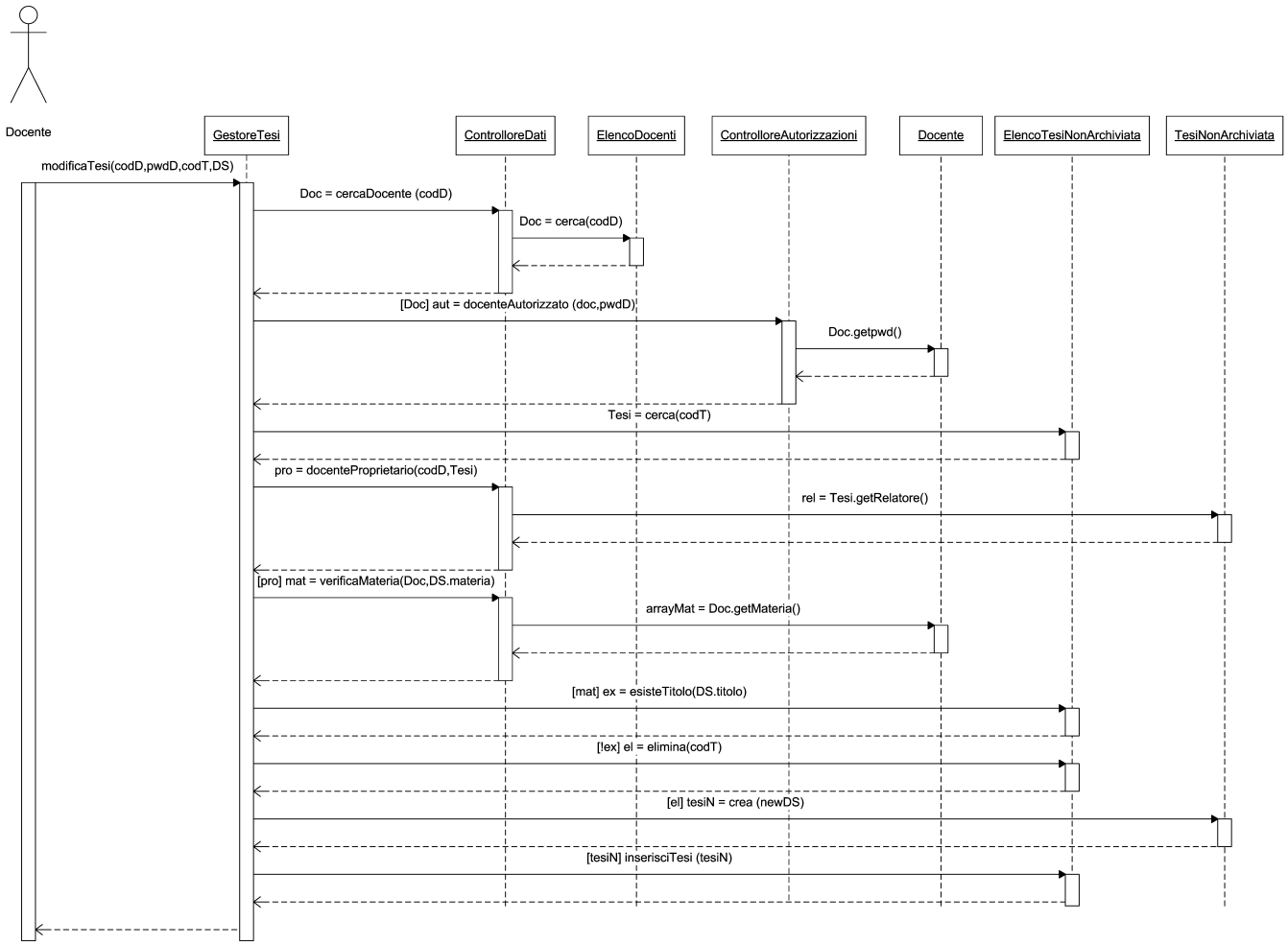


Figura 10.3: Sequence caso d'uso 2

Il docente effettua una chiamata alla procedura *modificaTesi* della classe GESTORETESI, fornendo come parametri: il codice della tesi non ancora archiviata da modificare, un DataStream contenente le nuove informazioni da associare alla tesi scelta e i dati necessari all'autenticazione.

Da questo punto in poi sarà la classe GESTORE a verificare indirettamente autenticazione e permessi e a gestire l'eventuale modifica; l'utente, in questo ca-

so il docente, sarà avvisato solo al termine dell'esecuzione della buona o cattiva riuscita della modifica.

La classe GESTORETESI, dopo aver ricevuto l'istanza del docente dalla classe CONTROLLOREDATI, verifica tramite una chiamata a CONTROLLOREAUTORIZZAZIONI i dati relativi all'autenticazione.

Il primo controllo che viene richiesto al sistema è, come da specifiche, verificare che il docente sia proprietario della proposta di tesi che ha indicato: non deve infatti essere possibile la modifica di proposte di tesi di altri professori.

La serie di controlli successivi invocati da GESTORETESI verificano i dati presenti nel DataStream in ingresso. Non è possibile modificare tutti i dati relativi ad una proposta di tesi: ad esempio non è possibile modificare il relatore, la sessione o lo studente. Per quest'ultima modifica infatti è necessario ricondursi al caso d'uso 5 (assegnazione di tesi). Nell'elenco delle tesi non archiviate sono presenti sia tesi assegnate che tesi libere, le modifiche offerte dal sistema con queste procedura devono tener conto dei dati principali relativi alla proposta di tesi: la materia e il titolo di tesi.

La classe GESTORETESI opportunamente richiama i metodi necessari per verificare che le informazioni presenti nel DataStream, se decodificate correttamente, non creino ambiguità o violino le specifiche progettuali, in particolare:

- un docente è autorizzato a proporre tesi relativamente ai propri insegnamenti e corsi
- non possono esistere due Tesi non ancora archiviate con lo stesso titolo

è interessante notare come il sistema non si limiti ad associare una sola materia ad ogni docente: la classe CONTROLLOREDATI infatti si preoccupa di recuperare una lista di materie associate al docente e controllare se tra esse figura la materia che si vuole attribuire alla proposta di tesi da modificare.

Una volta superati tutti i controlli, la classe GESTORETESI acquisisce i dati necessari per la creazione di una nuova proposta di tesi giustapponendo i dati in ingresso forniti dal docente con quelli relativi alla tesi da modificare creando un nuovo DataStream (newDS). Quest'operazione è necessaria per evitare la perdita di dati e per supportare la scelta di creare un sottoinsieme di proprietà non modificabili per ogni tesi.

Dopo aver correttamente ottenuto questo nuovo DataStream, la tesi da modificare viene prima cancellata tramite la chiamata *Elimina* della classe ELENCOTESINONARCHIVIATE, poi ricostruita tramite il metodo *crea* della classe TESINONARCHIVIATA e successivamente inserita nell'elenco con l'opportuna chiamata.

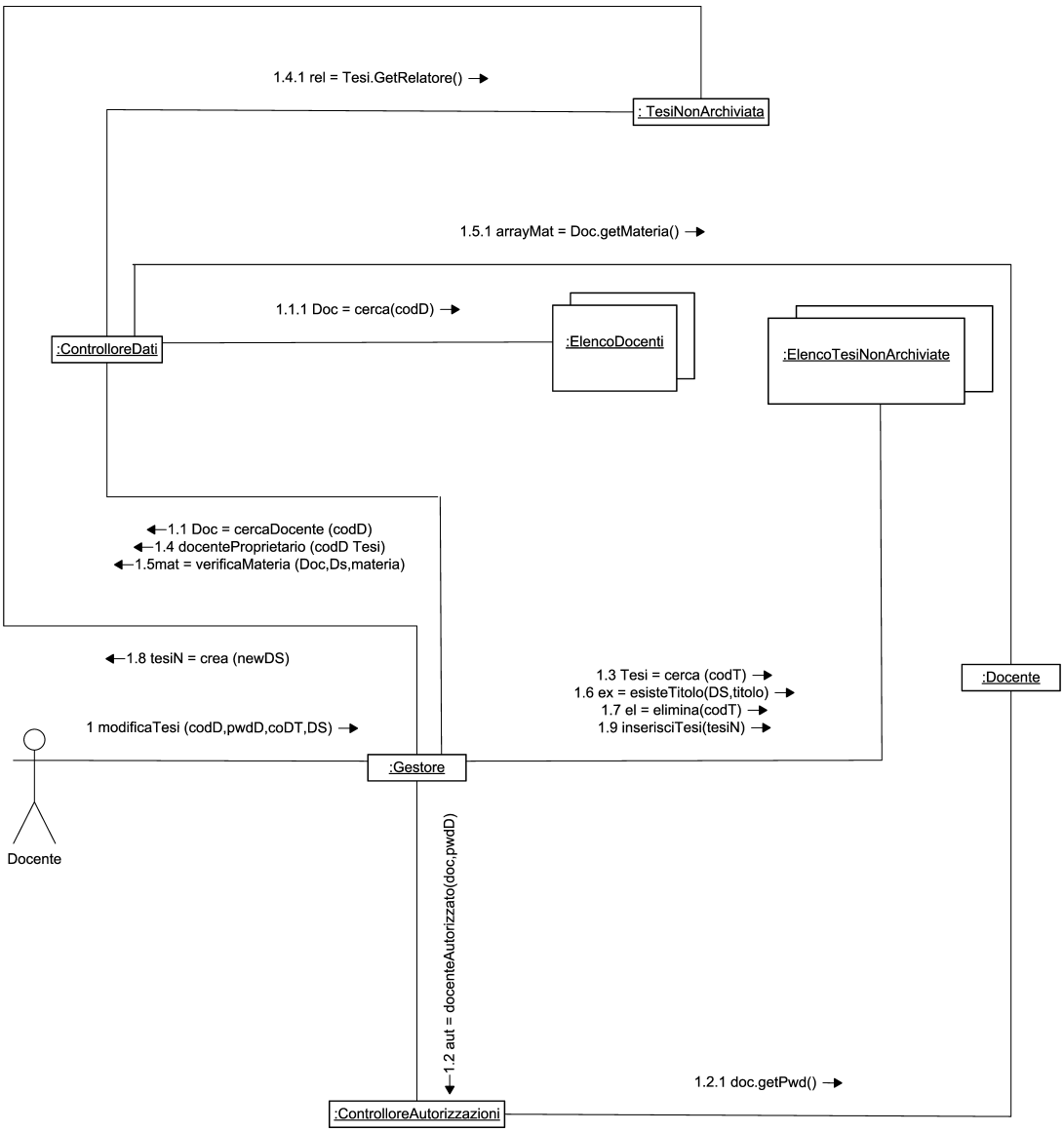


Figura 10.4: Diagramma di Collaborazione del caso d'uso 2

Nel diagramma di collaborazione è possibile seguire, tramite i numeri progessivi, lo sviluppo delle chiamate e l'uso dei canali di comunicazione.

10.3 Il caso d'uso 3

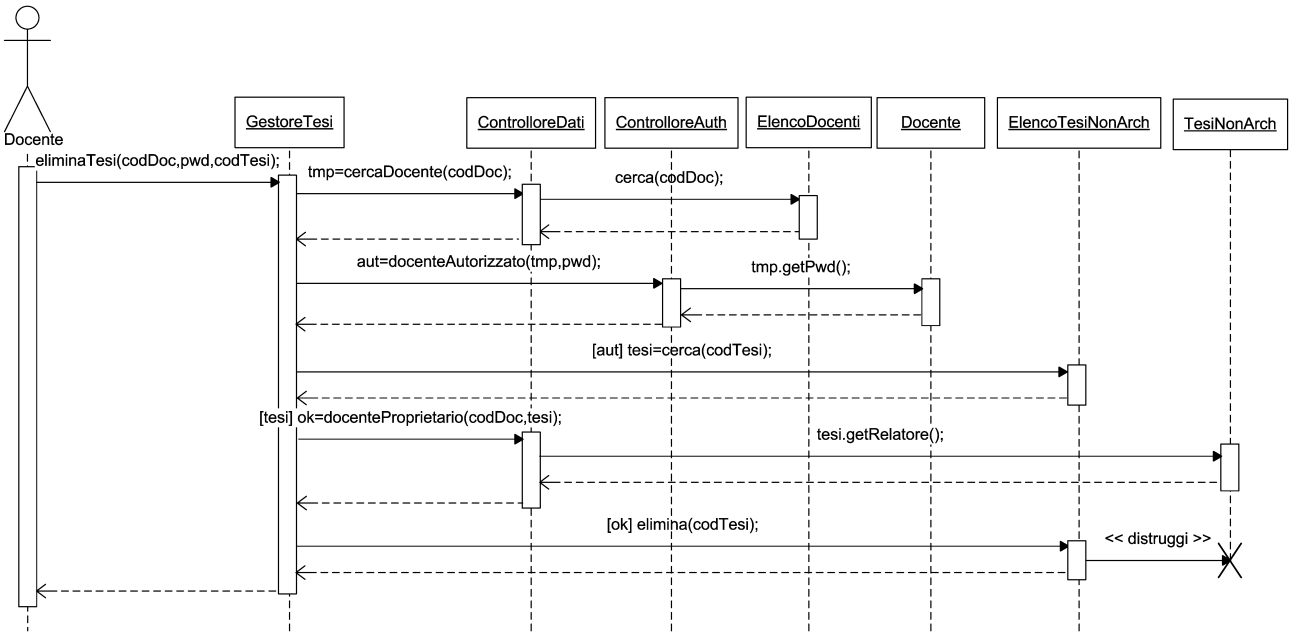


Figura 10.5: Sequence caso d'uso 3

Il caso d'uso ha inizio con l'invocazione del metodo *eliminaTesi* della classe GESTORE da parte del docente. I parametri forniti sono: UserID del docente, password, e il codice della tesi da eliminare. A questo punto il controllo passa al GESTORE, il quale effettuerà tutti i controlli necessari ad accertare che l'eliminazione sia lecita. Prima di tutto verifica le credenziali di autorizzazione fornite dal docente e lo fa in maniera del tutto analoga agli altri casi d'uso, affidandosi alle funzionalità fornite dalla classe CONTROLLOREAUTENTICAZIONE. Se il login ha buon esito (ovvero il docente esiste e la password fornita è corretta), il GESTORE procede verificando l'effettiva esistenza della tesi che si desidera eliminare tramite una chiamata al metodo *Cerca*(codiceTesi) di ELENCOTESINONARCHIVIATE.

Se la tesi esiste, si procede verificando che il docente che ne richiede l'eliminazione sia effettivamente quello che l'ha pubblicata. Questo si ottiene chiamando il metodo *docenteProprietario* della classe CONTROLLOREDATI, passando come parametri il codice del docente e l'istanza della tesi appena trovata. Per effettuare il controllo, la classe CONTROLLOREDATI ottiene il valore dell'attributo Relatore dell'istanza di TESI chiamando il metodo *getRelatore* e lo confronta con il codiceDocente ricevuto come parametro. Se anche questo controllo ha buon fine, il metodo *docenteProprietario* ritorna TRUE a GESTORE, il quale, riottenuto il controllo, effettua l'eliminazione invocando il metodo elimina della classe ELENCOTESINONARCHIVIATE. In caso contrario, l'eliminazione non verrà effettuata segnalando l'errore al docente.

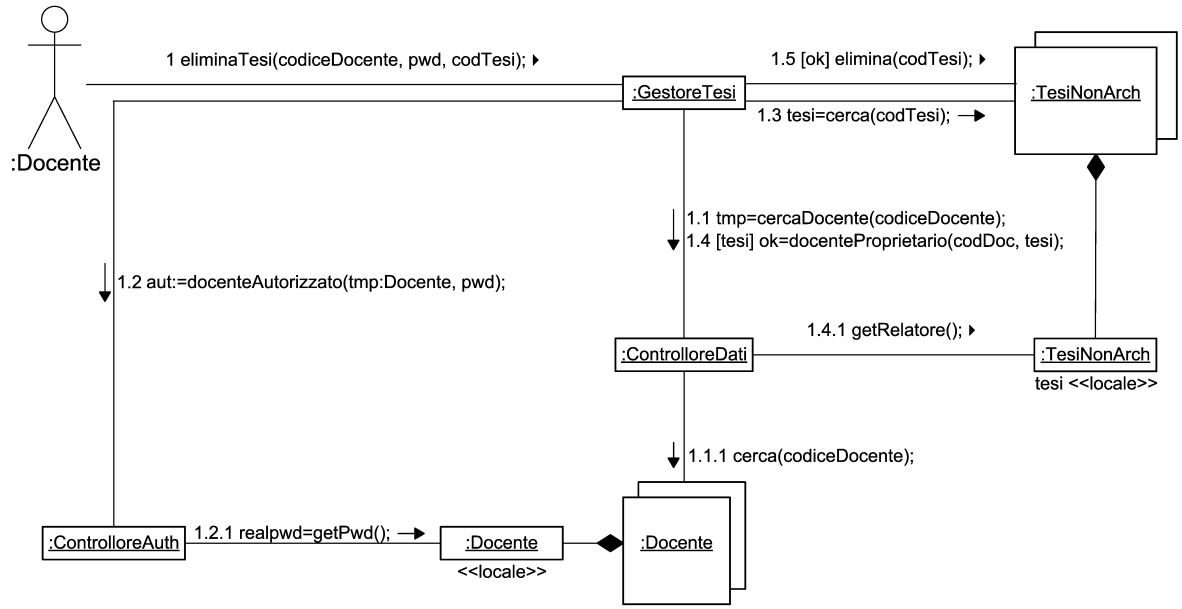


Figura 10.6: Diagramma di collaborazione del caso d'uso 3

10.4 Il caso d’uso 4

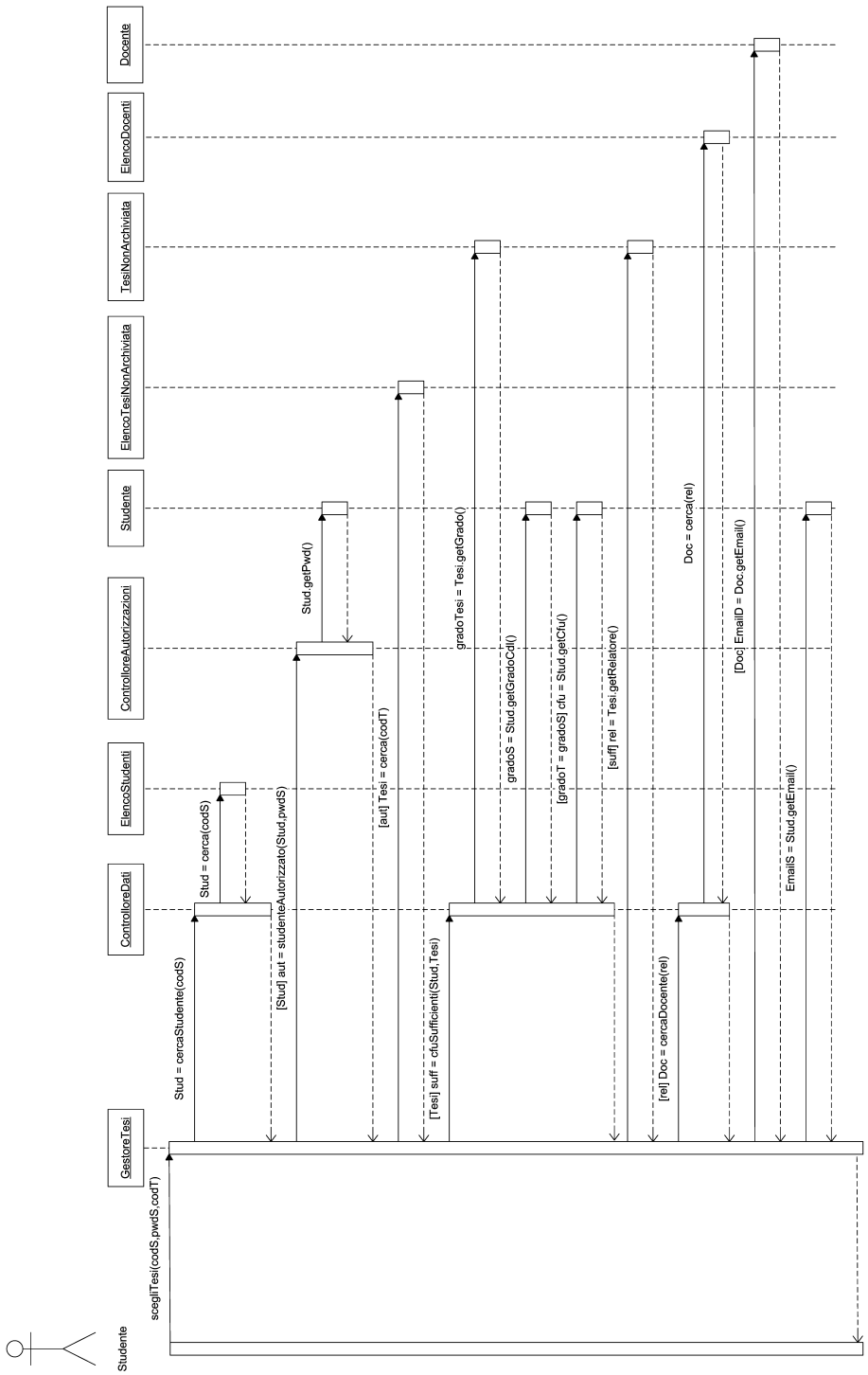


Figura 10.7: Sequence caso d’uso 4

Lo studente effettua una chiamata al metodo *scegliTesi* della classe GESTORETESI fornendo come parametri il codice della tesi a cui è interessato e i dati necessari all'autenticazione. Nel caso in cui lo studente sia in possesso delle qualità necessarie per richiedere quella tesi, verrà spedita una comunicazione al docente interessato per avvisarlo di un possibile colloquio futuro. La classe GESTORETESI, dopo aver ricevuto l'istanza dello studente dalla classe CONTROLLOREDATI, verifica i dati immessi relativi all'autenticazione. Per prima cosa il sistema si occupa di recuperare le informazioni relative alla tesi che lo studente ha scelto, per determinarne il grado (triennale, specialistica, quinquiennale), la disponibilità e altre informazioni satellite non strettamente necessarie per l'eventuale assegnazione. Ci sono regole ben precise che legano la possibilità di richiedere l'assegnazione di una tesi: al fine di evitare spiacevole inconvenienti il sistema controlla se lo studente sia o meno in possesso dei crediti formativi (CFU) necessari senza preoccuparsi di verificare se tale studente sia già assegnatario di un'altra proposta di tesi. Questa scelta, in principio contestata e molto discussa, non limita le possibilità di consultazione e richiesta di informazioni per studenti già assegnatari di una tesi. Il controllo necessario per evitare che uno studente risulti assegnatario di più proposte di tesi è presente nel caso d'uso 5 (assegnazione di una tesi): ciò che si intende permettere al sistema applicando questa politica è dare la possibilità di sostenere un colloquio informativo riguardo una tesi, anche a coloro che, in teoria, stanno già sviluppandone una. Nella pratica questa scelta modella il caso in cui uno studente decida di cambiare tesi qualora ne trovi una libera che più gli piace.

La fase centrale di questo caso d'uso sta nel controllo delle credenziali dello studente in termini di cfu: qualunque siano le politiche e le metriche di scelta (150 cfu per potere richiedere una tesi triennale, 80 per poter richiedere una tesi specialistica o quinquiennale) il sistema tenta di essere il più flessibile e scalabile possibile, controllando (tramite informazioni non necessariamente intrinseche a questo progetto) l'associazione tra il tipo di tesi e il grado dello studente.

Ciò che si vuole evitare è che uno studente della laurea triennale, in possesso di almeno 150 crediti, richieda un colloquio per una tesi pensata per una laurea specialistica (attenzione, senza questo controllo lo studente risulterebbe avere

il numero di crediti necessari) o che uno studente della laurea triennale in possesso di 100 crediti (non potendo sviluppare un tesi triennale) richieda una tesi per la laurea quinquiennale. Il caso simmetrico, per come sono definite attualmente le specifiche, non può accadere: uno studente della laurea specialistica non sarà mai in possesso di 150 crediti prima della consegna della tesi; se in ogni caso queste politiche dovessero cambiare, dovessero cambiare le regole dei corsi di laurea o i punti assegnati ad ogni esame, il nostro sistema si rivelerebbe facilmente riconfigurabile e manutenibile.

Con questa scelta non intendiamo impedire colloqui informativi tra docenti e tesisti che per qualche motivo vorrebbero vedersi assegnata un tesi non idonea al loro grado di studio (bisogna considerare che non necessariamente una proposta di tesi sia specificata o non sia adattabile alle richieste degli studenti), a fronte di un colloquio il docente interessato deciderà se modificare o meno il grado della proposta di tesi, dando quindi possibilità ad altri studenti di richiederne l'assegnazione.

Nel caso in cui tutti i controlli siano andati a buon fine, al docente viene spedita un'Email in cui viene indicato anche l'indirizzo Email dello studente per una eventuale risposta.

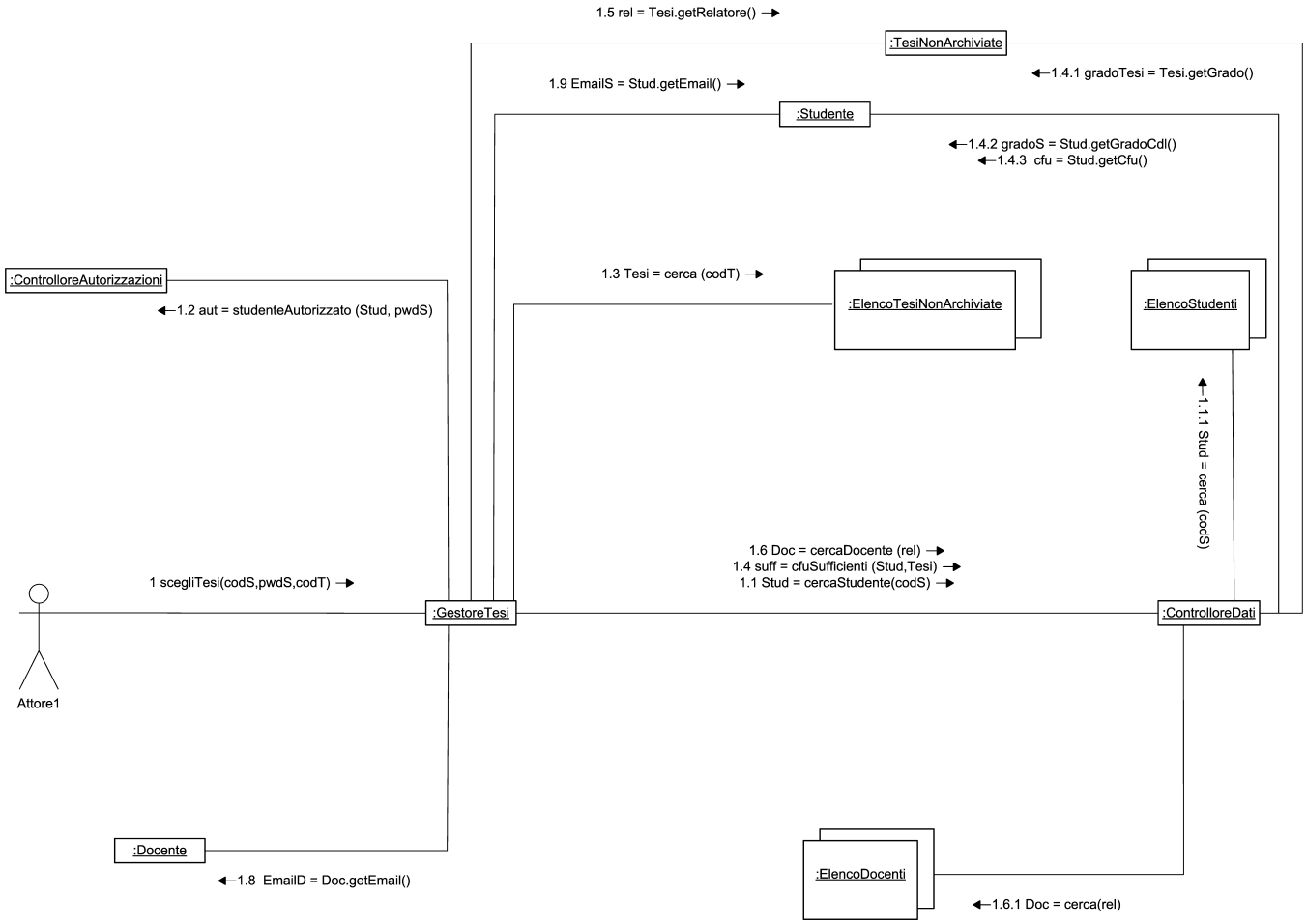


Figura 10.8: Diagramma di collaborazione del caso d'uso 4

Dal diagramma di collaborazione si può notare, oltre al numero di messaggi e la sequenza delle chiamate, l'importanza centrale della classe GESTORETESI e l'intensa collaborazione con la classe *ControlloreDati*.

10.5 Il caso d’uso 5

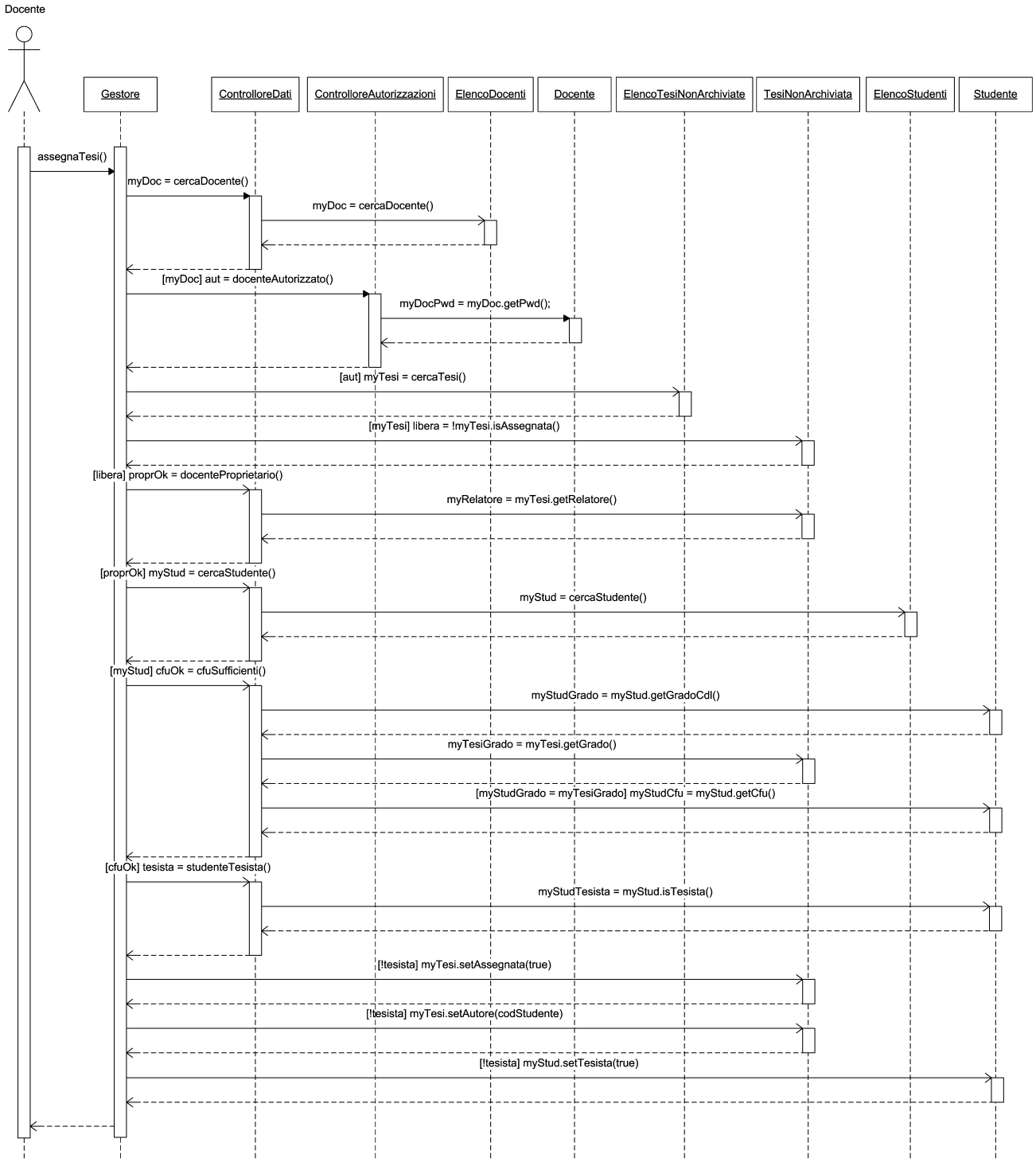


Figura 10.9: Sequence caso d’uso 5

Un docente invoca il metodo *AssegnaTesi* di GESTORE per effettuare l'assegnazione di una tesi ad uno studente. Di conseguenza i parametri necessari sono il codice identificativo del docente (codDocente), la relativa password, il codice della tesi che deve essere assegnata ed il codice dello studente. Il GESTORE acquisisce il controllo delle operazioni da svolgere e ritornerà al docente un valore booleano al termine dell'esecuzione. Il primo passo consiste nell'autenticazione del docente, perciò la prima operazione eseguita è una chiamata del GESTORE al metodo *cercaDocente* della classe CONTROLLOREDATI per ottenere una istanza del docente su cui effettuare i controlli. Questa chiamata è reinviata dal CONTROLLOREDATI alla classe ELENCODOCENTI che provvede ad effettuare la ricerca dell'istanza del docente e la ritorna al metodo chiamante. Una volta ottenuta l'istanza del docente il GESTORE deve verificare l'autenticazione e per farlo invoca il metodo *docenteAutorizzato* del CONTROLLOREAUTORIZZAZIONI passando come parametri l'istanza del docente e la password da lui fornita. Il CONTROLLOREAUTORIZZAZIONI chiama il metodo *getPwd* sulla classe DOCENTE passandogli l'istanza ed ottiene una stringa. A questo punto il controllore può procedere al confronto tra la stringa ottenuta e quella passata come parametro dal docente, e nel caso siano uguali ritorna il controllo al GESTORE passandogli un valore booleano positivo.

Il GESTORE a questo punto procede nel verificare la disponibilità di assegnazione della tesi, perciò inizialmente deve ottenerne l'istanza. GESTORE chiama dunque il metodo *cercaTesi* su ELENCOTESINONARCHIVIATE ed in caso che questa esista ne ottiene come valore di ritorno l'istanza. Avuta l'istanza deve accertarsi che la tesi sia effettivamente assegnabile e perciò invoca la chiamata *isAssegnata* alla classe *TesiNonArchiviata* verificando che il valore booleano di ritorno sia falso. Fatto questo il sistema deve verificare che il docente sia l'effettivo proprietario della tesi e che possa dunque essere autorizzato ad assegnarla. Per poterlo fare il GESTORE invoca il metodo *docenteProprietario* con parametri codice docente e tesi. Il CONTROLLOREDATI effettua il controllo effettuando la chiamata *getRelatore* alla classe TESINONARCHIVIATA e confrontando il dato ottenuto con quello passato come parametro dal GESTORE. In caso di esito po-

sitivo il GESTORE può quindi verificare i dati dello studente per controllare che sia possibile assegnarli la tesi. Effettua la CERCASTUDENTE presso il CONTROLLOREDATI, che come al solito chiama la cerca con parametro codice studente alla classe ELENCOSTUDENTI. Con l'istanza il GESTORE può verificare che lo studente abbia i crediti sufficienti per poter ottenere l'assegnazione della tesi. GESTORE chiama dunque la *cfuSufficienti* del CONTROLLOREDATI ed attende il valore booleano di ritorno. Il controllore deve controllare la tipologia di corso a cui è iscritto lo studente e confrontarla con quella della tesi proposta. Per fare ciò chiama *getGradoCdl* alla classe STUDENTE, *getGrado* alla classe TESINONARCHIVIATA e ne confronta i risultati. Se i valori di ritorno coincidono allora esegue una chiamata *getCfu* alla classe STUDENTE per ottenere i crediti posseduti dallo studente. Un metodo automatico del sistema deve determinare se i crediti sono sufficienti per ottenere l'assegnazione, confrontandoli con il valore minimo associato alla tipologia della tesi. A questo punto, se anche questo controllo va a buon fine, il GESTORE deve verificare come ultima cosa che lo studente non sia già assegnatario di un'altra tesi. Chiama il metodo *studenteTesista* della classe CONTROLLOREDATI, il quale, utilizzando come parametro l'istanza dello studente effettua la chiamata *isTesista* alla classe STUDENTE.

Se lo studente non è già tesista, il GESTORE può dunque procedere alla fase di assegnazione. Innanzitutto chiama la funzione *setAssegnata* alla classe TESINONARCHIVIATA per dichiarare la tesi assegnata. Se l'operazione ha successo GESTORE chiama *setAutore* con parametro codice studente alla classe TESINONARCHIVIATA. Come ultimo passaggio gestore chiama il metodo *setTesista* alla classe STUDENTE per dichiarare che lo studente è diventato assegnatario.

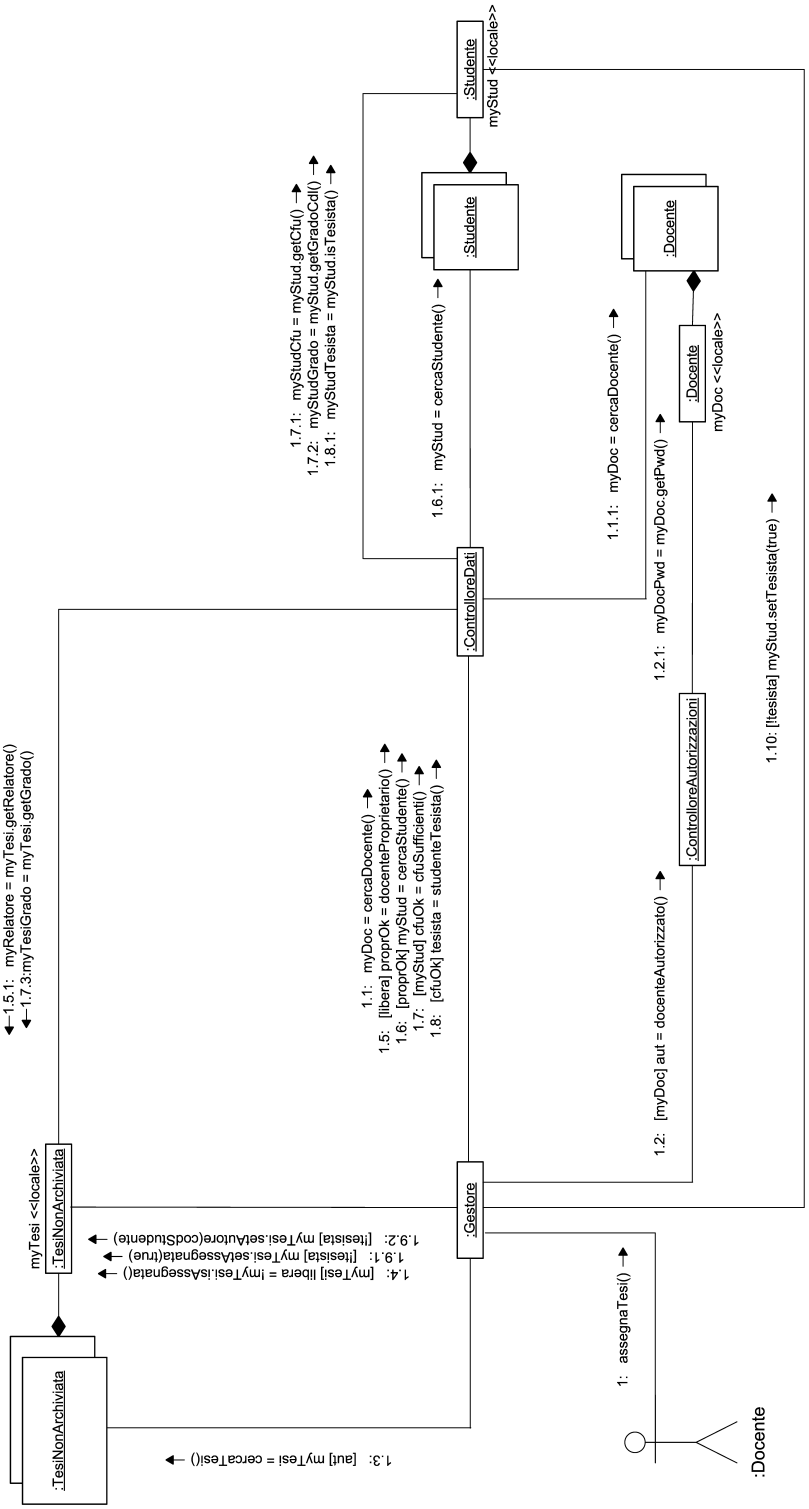


Figura 10.10: Diagramma di collaborazione del caso d’uso 5

10.6 Il caso d’uso 6

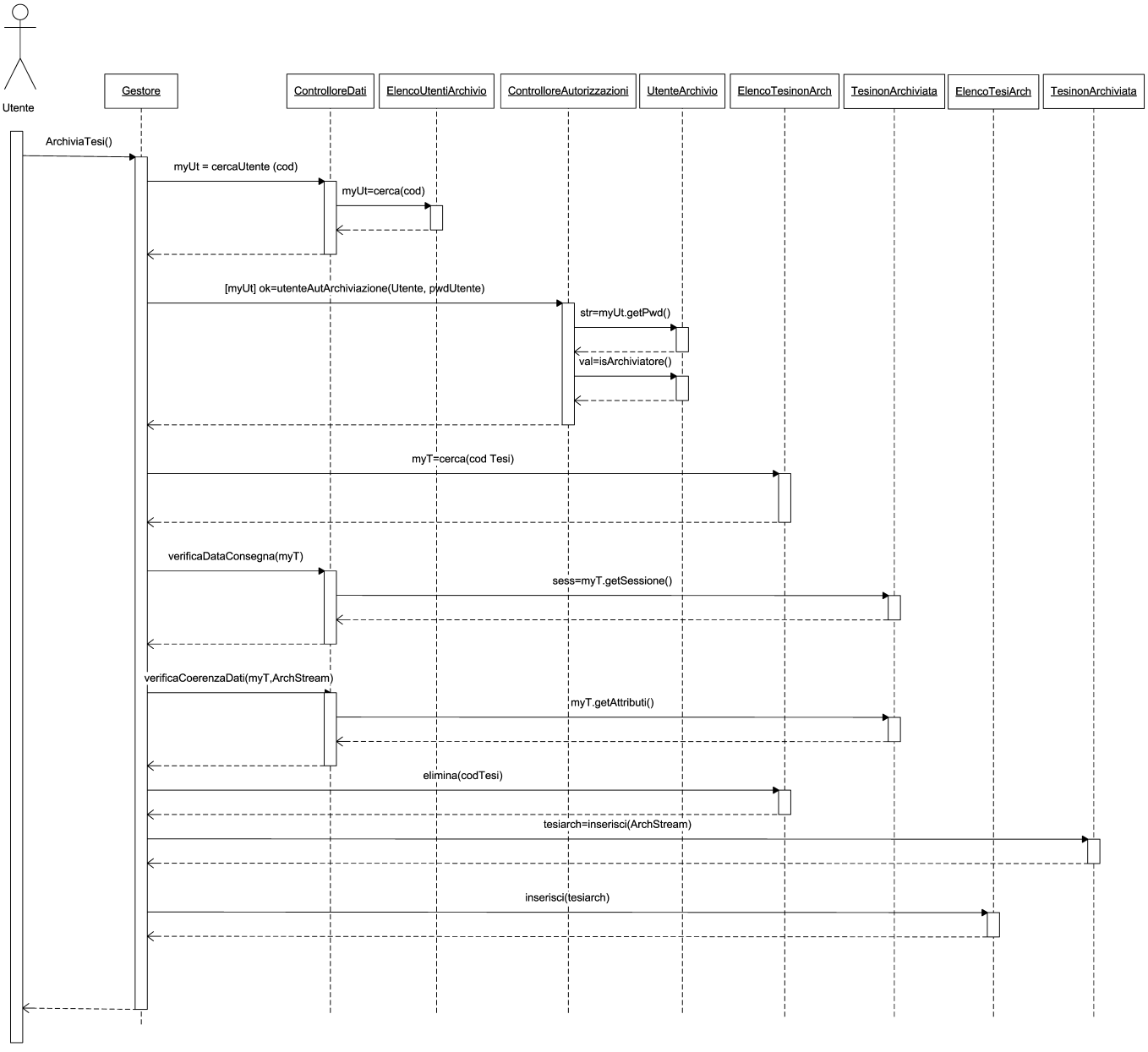


Figura 10.11: Sequence caso d’uso 6

L'utente invoca il metodo *ArchiviaTesi* di GESTORE per effettuare l'archiviazione di una tesi. Tale metodo prende come parametri il codice identificativo dell'utente (codUtente), la relativa password, il codice della tesi che deve essere archiviata ed uno Stream di dati che devono essere verificati ed inseriti tra gli attributi della tesi una volta archiviata. Il GESTORE acquisisce il controllo delle operazioni da svolgere e ritornerà all'utente un valore booleano al termine dell'esecuzione. Il primo passo consiste nell'autenticazione dell'utente, perciò la prima operazione eseguita è una chiamata del GESTORE al metodo *cercaUtente* della classe CONTROLLOREDATI per ottenere una istanza dell'utente su cui effettuare i controlli. Questa chiamata è forwardata dal CONTROLLOREDATI alla classe ELENCOUTENTIARCHIVIO che provvede ad effettuare la ricerca dell'istanza dell'utente e la ritorna al metodo chiamante. Una volta ottenuta l'istanza dell'utente il GESTORE deve verificare l'autenticazione e per farlo invoca il metodo *utenteAutArchiviazione* del CONTROLLOREAUTORIZZAZIONI passando come parametri l'istanza dell'utente e la password da lui fornita. Il CONTROLLOREAUTORIZZAZIONI chiama il metodo *getPwd* sulla classe UTENTEARCHIVIO passandogli l'istanza ed ottiene una stringa. A questo punto può procedere al confronto tra la stringa ottenuta e quella passata come parametro dall'utente. Se questo controllo va a buon fine il controllore deve verificare che l'utente sia autorizzato ad inserire tesi in Archivio. Per fare questo chiama il metodo *isArchiviatore* alla classe UTENTEARCHIVIO e verifica il valore booleano di ritorno. Se anche questo controllo va a buon fine, il CONTROLLOREAUTORIZZAZIONI ritorna il controllo al GESTORE passandogli un valore booleano positivo.

Il GESTORE a questo punto, in caso di positiva autenticazione dell'utente, deve verificare i dati relativi alla Tesi da archiviare. Come primo passaggio deve accertarsi che la Tesi sia effettivamente esistente, e perciò invoca la chiamata cerca alla classe ELENCOTESINONARCHIVIATE passandogli il codice della Tesi. ELENCOTESINONARCHIVIATE effettua la ricerca e restituisce, in caso ci sia, l'istanza della tesi. Il GESTORE può quindi verificare se i dati per l'archiviazione sono corretti, perciò chiede il controllo della data di consegna alla classe CONTROLLOREDATI utilizzando il metodo *verificaDataConsegna*. Il CONTROLLOREDATI

procede quindi a richiedere la sessione della tesi con *getSessione* a TESINONARCHIAVIATA. La stringa ottenuta come valore di ritorno è passata come parametro ad un metodo che abbiamo considerato implementato dal sistema. Tale metodo deve confrontare la data corrente con la data di consegna associata alla sessione della tesi e ritornare TRUE se la prima risulta inferiore. Tale valore corrisponde al fatto che la consegna sia in tempo rispetto alla deadline della sessione. Se tale controllo dà esito positivo il GESTORE procede alla verifica dei dati complementari immessi dall'utente. Chiama dunque la *verificaCoerenzaDati* passando come parametri lo stream dei dati e l'istanza della tesi da controllare. Il controllo passa al CONTROLLOREDATI che in sequenza chiama i metodi per ottenere gli attributi della tesi alla classe TESINONARCHIVIATA. Questi sono *getAutore*, *getRelatore*, *getMateria*, ecc. Per brevità e leggibilità del diagramma di sequenza questi metodi sono stati raggruppati in un'unica chiamata fittizia *getAttributi*. Per ogni attributo così ottenuto il CONTROLLOREDATI effettua la verifica che i parametri dello Stream siano coerenti. Se tutti i dati coincidono l'operazione ritorna al GESTORE un booleano positivo, negativo altrimenti. Il GESTORE in caso di esito positivo può quindi procedere alla fase di archiviazione della tesi. Come prima operazione chiama il metodo *eliminaTesi* della classe ELENCOTESINONARCHIVIATE passando come parametro il codice della Tesi. Una volta ritornato il controllo effettua la chiamata di inserisci su TESIARCHIVIATE passandogli l'intero stream di dati. TESIARCHIVIATE, una volta inserita la tesi ne ritorna l'istanza. Con questa istanza il GESTORE può chiamare l'inserimento nell'elenco delle tesi archiviate. Il metodo è *Inserisci* e l'unico parametro è l'istanza.

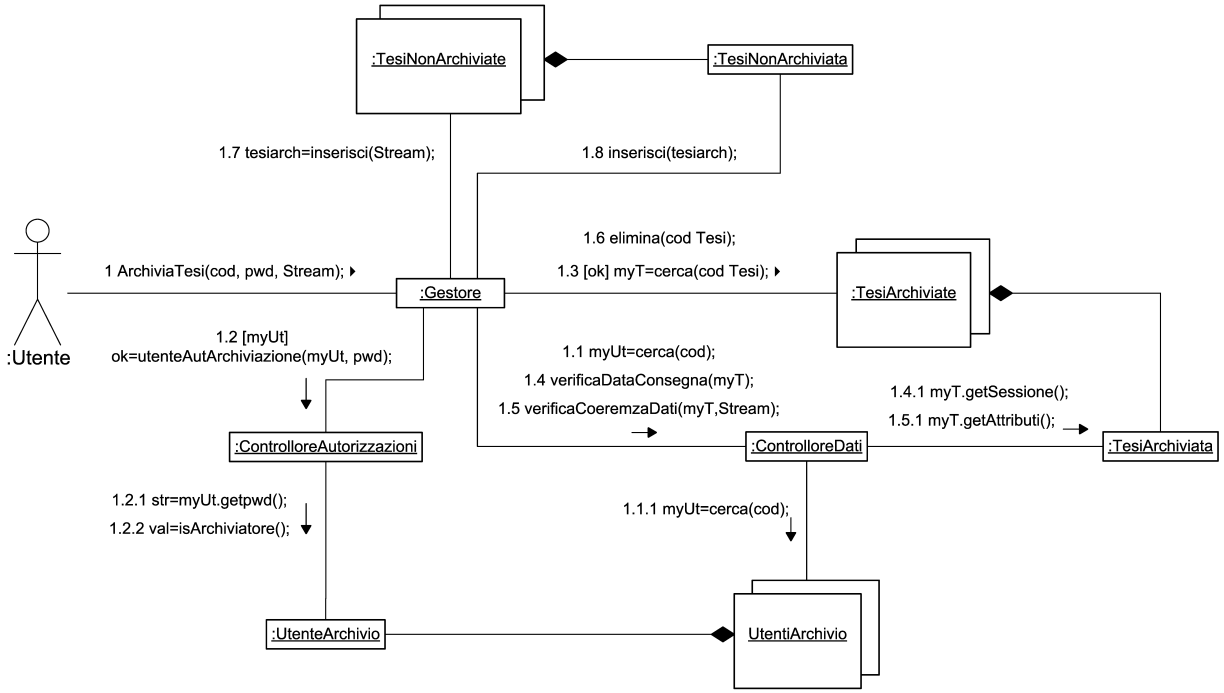


Figura 10.12: Diagramma di collaborazione del caso d’uso 6

10.7 Il caso d'uso 7

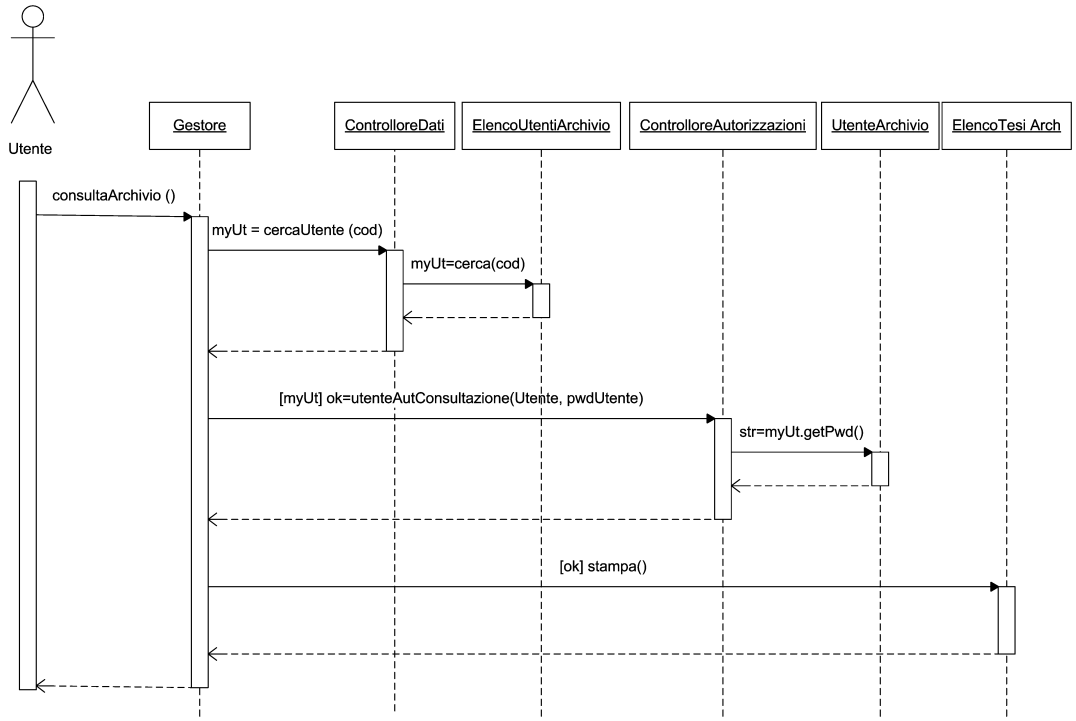


Figura 10.13: Sequence caso d'uso 7

Questo caso d'uso riguarda la consultazione dell'archivio delle tesi da parte di utenti autorizzati. La sequenza delle operazioni inizia quando l'utente invoca il metodo *ConsultaArchivio* della classe GESTORE. Tale metodo prende come parametri il codice identificativo dell'utente (codUtente) e la corrispondente password, necessari per poter verificare se l'utente sia o meno autorizzato ad effettuare tale operazione. Il primo passo consiste nell'autenticazione dell'utente, perciò la prima operazione eseguita è una chiamata del GESTORE al metodo *cercaUtente* della classe CONTROLLOREDATI per ottenere una istanza dell'utente su cui effettuare i controlli. Questa chiamata è forwardata dal CONTROLLOREDATI alla classe ELENCOUTENTIARCHIVIO che provvede ad effettuare la ricerca dell'istanza dell'utente e la ritorna al metodo chiamante. Una volta ottenuta l'istanza

dell'utente il GESTORE deve verificare l'autenticazione e per farlo invoca il metodo *utenteAutConsultazione* del CONTROLLOREAUTORIZZAZIONI passando come parametri l'istanza dell'utente e la password da lui fornita. Il CONTROLLOREAUTORIZZAZIONI chiama il metodo *getPwd* sulla classe UTENTEARCHIVIO passandogli l'istanza ed ottiene una stringa. A questo punto può procedere al confronto tra la stringa ottenuta e quella passata come parametro dall'utente. Se questo controllo va a buon fine il controllore ritorna il controllo al GESTORE passandogli un valore booleano positivo, negativo nel caso opposto. Il GESTORE a questo punto, in caso di positiva autenticazione dell'utente, procede invocando un metodo *stampa* della classe ELENCOTESIARCHIVIATE, che cicla su tutti gli elementi dell'archivio tramite il metodo *prossimo*. La funzione è void quindi il metodo non ha valore di ritorno per l'utente.

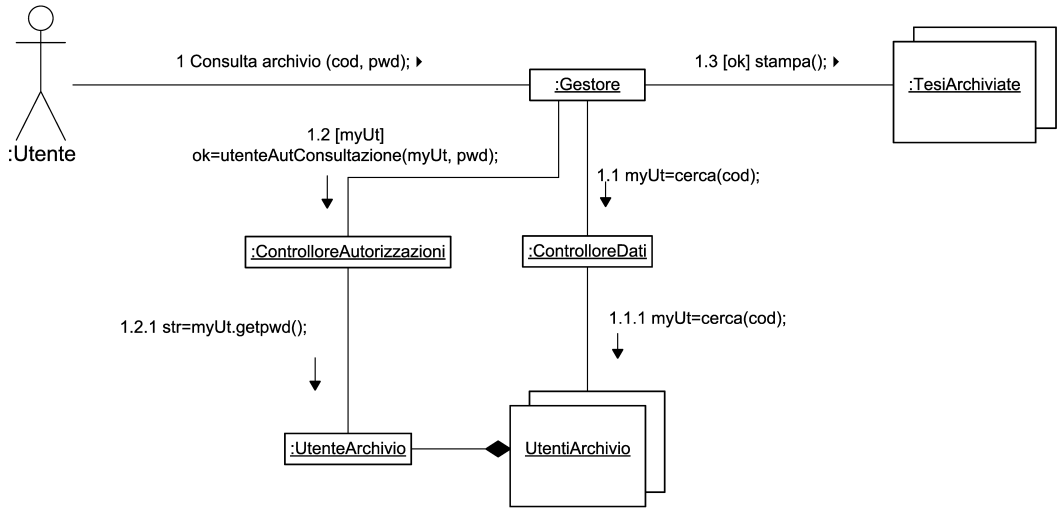


Figura 10.14: Diagramma di collaborazione del caso d'uso 7

Capitolo 11

Diagrammi degli stati

In questa parte si vuole mettere in evidenza un altro aspetto dinamico del sistema, dopo quello già messo in evidenza con i diagrammi delle attività. Questi ultimi sono infatti "un tipo speciale di diagrammi di stato in cui gli stati sono di azione o di sottoattività, e le transizioni sono attivate automaticamente al completamento delle azioni e delle attività di uno stato. La differenza sostanziale tra i diagrammi della attività e i diagrammi di stato risiede nello scopo delle due modellazioni: mentre per i primi interessa modellare i processi di business in cui partecipano diversi oggetti, per i secondi interessa solitamente modellare il ciclo di vita di un’unica entità reattiva facente parte del sistema stesso. Giova ricordare, facendo riferimento a [UML], che un’entità reattiva:

- risponde ad eventi esterni (che nascono al di fuori del contesto dell’entità);
- ha un ciclo di vita definito, modellabile come una serie di stati, transizioni, eventi;
- ha un comportamento corrente che dipende dai comportamenti precedenti.

Anche se "nella modellazione OO le macchine a stati possono essere utilizzate per modellare il comportamento dinamico di sistemi interi, sottosistemi, casi d’uso e classi", "sono solitamente utilizzate per modellare il comportamento dinamico di classi". In forza di questo concetto ribadito più volte in [UML], abbiamo deciso di creare diagrammi di stato solamente per le due classi più interessanti dal punto

di vista del comportamento dinamico e del ciclo di vita, oltre ad un diagramma a livello di sistema, utile per fornire una visione "globale" e trasversale a tutti i casi d’uso del comportamento dello stesso. Abbiamo quindi realizzato diagrammi di stato per:

- la classe tesi;
- la classe studente;
- l’intero sistema;

Evitando di realizzare un diagramma di stato per i vari sottosistemi o per classi prive di particolare rilevanza dal punto di vista del comportamento dinamico.

11.1 Diagrammi degli stati a livello di sistema

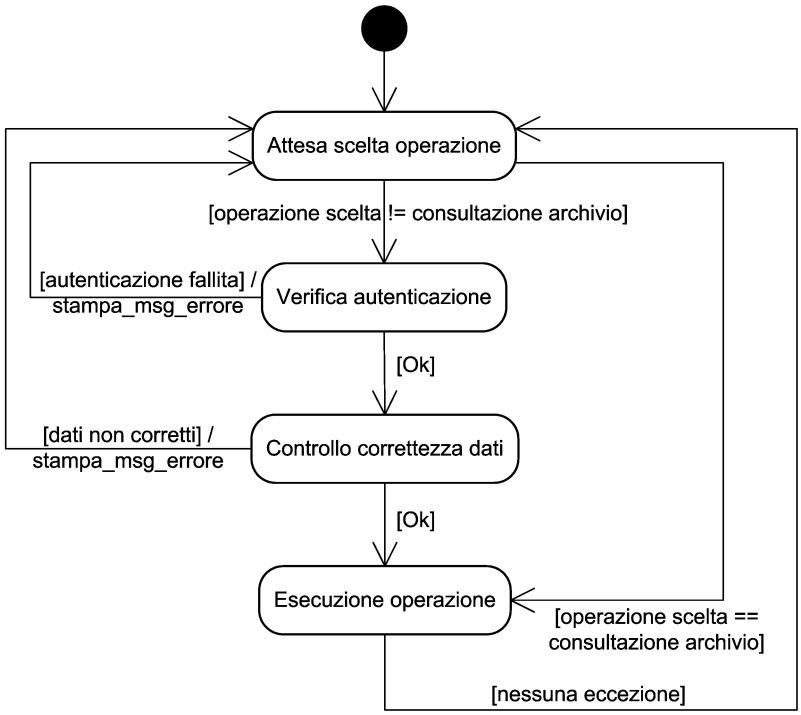


Figura 11.1: Stati del sistema

Appena inizializzato, il sistema si trova in uno stato di "Attesa", in cui si attende la scelta da parte dell'utente di una delle operazioni messe a disposizione dal sistema stesso. Appena viene scelta una operazione si passa allo stato di verifica delle credenziali di autenticazione, tranne nel caso in cui si scelga di consultare l'archivio delle tesi: in quest'ultimo caso infatti, come si può evincere anche dall'analisi dei requisiti, non è richiesto nessun controllo di autenticazione. Nel caso

di consultazione dell'archivio si passa direttamente allo stato di "Esecuzione dell'operazione", per poi tornare allo stato di attesa. Per tutte le altre operazioni, se l'autenticazione va a buon fine si passa allo stato di controllo di correttezza dei dati inseriti dall'utente. Questo stato ha lo scopo di verificare la leicità dell'operazione che si tenta di svolgere e la correttezza formale dei dati forniti per portarla a termine. Nel caso in cui tale controllo dovesse fallire, come nel caso di fallimento della procedura di autenticazione, si ritorna allo stato di attesa, stampando un opportuno messaggio di errore all'utente. Superato anche il controllo di correttezza dei dati, il sistema giunge allo stato di esecuzione dell'operazione, in cui banalmente viene portato a termine il compito richiesto per poi ritornare allo stato di attesa.

11.2 Diagrammi degli stati delle classi principali

11.2.1 La classe studente

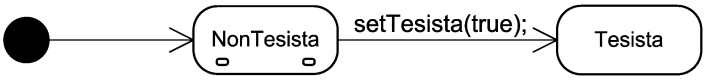


Figura 11.2: Diagramma degli stati dello studente non espanso

Questo diagramma degli stati mette in evidenza i due stati principali in cui si può trovare una qualsiasi istanza della classe studente:

- NonTesista : stato composito in cui si trova uno studente al quale non è ancora stata assegnata una tesi;
- Tesista : stato semplice in cui si trova uno studente dal momento in cui gli è stata assegnata una tesi.

La transizione dallo stato NonTesista allo stato Tesista avviene nel momento in cui viene chiamato il metodo setTesista(true); della classe Studente. Questa chiamata avviene durante il caso d'uso di assegnazione, nel caso in cui tutti i controlli tesi a verificare la leicità dell'assegnazione vadano a buon fine.

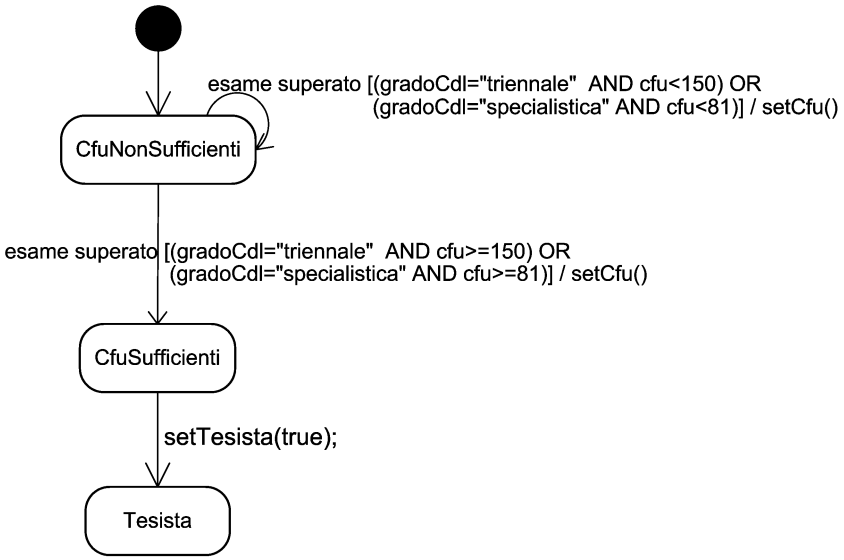


Figura 11.3: Diagramma degli stati dello studente espanso

Questo diagramma è equivalente al precedente, differendo da esso per la sola espansione dello stato composto NonTesista in due stati elementari:

- CfuNonSufficienti: lo studente non ha ancora i crediti sufficienti a inoltrare una domanda di prova finale;
- CfuSufficienti: lo studente ha acquisito i crediti sufficienti ad inoltrare domanda di prova finale.

La transizione dal primo al secondo stato avviene nel momento in cui uno studente passa un esame, e i crediti appena ottenuti gli permettono di superare la soglia minima necessaria per inoltrare domanda. Di conseguenza finchè uno studente si trova nello stato cfuNonSufficienti non potrà effettuare la transizione allo stato Tesista, e questo è coerente con le specifiche dei requisiti e con i vincoli della realtà di interesse. Una volta che lo studente si trova nello stato CfuSufficienti, è libero di fare domanda di prova finale (consultando l'elenco di prove finali e scegliendone una, causando l'invio automatico di una Email al docente). Successivamente, a seguito di un colloquio con esito positivo con il docente, lo studente

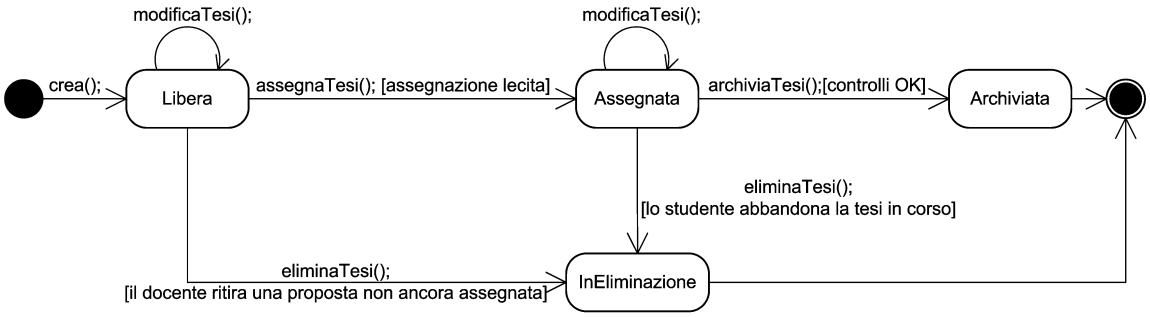


Figura 11.4: Stati di Tesi

vedrà assegnarsi la tesi e quindi, con la chiamata a setTesista(true), passerà allo stato "Tesista".

11.2.2 La classe Tesi

Per quanto riguarda gli stati assumibili dall'entità TESI, si evince chiaramente dal diagramma come, subito dopo la sua creazione, una Tesi si trovi nello stato "libera" che identifica una proposta di tesi pubblicata ma non ancora assegnata ad alcuno studente. Una tesi Libera diventa "Assegnata" nel momento in cui, a seguito di un colloquio positivo con uno studente interessato, il docente decide di assegnargli una prova finale. La transizione si completa solo a patto che tutti i controlli necessari vadano a buon esito. In alternativa, una tesi libera può effettuare una transizione allo stato "InEliminazione", nel momento in cui un docente decide di ritirarla chiamando il metodo eliminaTesi(); della classe Gestore. Da quel momento il ciclo di vita della tesi può considerarsi concluso (cfr. transizione da "InEliminazione" verso lo stato finale). Una Tesi Assegnata, passerà allo stato "Archiviata" nel momento in cui, una volta portata a termine, lo studente ne consegna con successo una copia in segreteria entro il termine ultimo fissato per la sessione corrente. Anche in questo caso il ciclo di vita della Tesi si considera concluso, in quanto non sono previste eliminazioni dall'archivio delle tesi svolte. Da notare che anche una Tesi "Assegnata" può transire allo stato "InEliminazione": ciò avviene nel caso in cui uno studente, una volta assegnatagli una certa tesi, si renda conto della eccessiva complessità che lo svolgimento della prova finale comporterebbe e decida quindi di rinunciare allo svolgimento della stessa. In tal

caso, a seguito di un colloquio con il relatore, il relatore stesso può decidere di accogliere la richiesta di rinuncia, eliminando la proposta di tesi ed eventualmente ripubblicandola. Infine, le transizioni da uno stato in se stesso sono abbastanza triviali e contemplano semplicemente i casi di modifiche apportate alla tesi a seguito della sua pubblicazione (errore nella digitazione del titolo o altro) o a seguito dell'assegnamento ad uno studente (i.e.: decisione/modifica della sessione i laurea in cui la tesi verrà discussa).

Parte IV

Deployment

Capitolo 12

Cenni generali sul Deployment

L'applicazione verrà sviluppata con un'architettura Enterprise JavaBeans, in particolare con l'application server open-source JBoss. L'architettura Enterprise JavaBeans fornisce un modello di componenti distribuiti per Java, e si basa sul concetto di componente EJB. Ognuno di questi componenti viene dislocato all'interno di un EJBcontainer, il quale fornisce al componente, in maniera del tutto automatizzata, una serie di servizi avanzati (distribuzione, transazioni, persistenza e sicurezza). Ogni componente EJB [UML, pagg. 346-348, 352-356] è costituito da quattro parti: 3 classi java e un file di descrizione XML. Vediamone la semantica in dettaglio al fine di comprendere meglio il diagramma di deployment ottenuto.

- **Interfaccia Home (HomeInterface)**: definisce l'interfaccia per cercare e/o creare istanze del componente EJB;
- **Interfaccia Remote (RemoteInterface)**: Definisce l'interfaccia dei metodi "di business" del componente;
- **Implementazione EJB**: Implementa le due interfacce precedenti (Home e Remote);
- **Descrittore di Deployment (Deployment Descriptor)**: File in formato XML contenente una descrizione, utile al container, di quali siano i servizi richiesti dal bean.

Si potrebbe quindi pensare che l'implementazione EJB implementi nel senso stretto del termine le due interfacce Remota e Home, ma in realtà queste ultime sono implementate rispettivamente da un oggetto EJB e da un oggetto Home generati automaticamente dal Contenitore EJB all'atto della sua attivazione. Nonostante questo, l'implementazione EJB deve comunque includere i metodi definiti nell'interfaccia remota e nell'interfaccia home, pena il fallimento del processo di compilazione dei file JAR per il client e per il server. Tutte le parti EJB vengono generalmente incluse in un unico file JAR che, al momento della sua dislocazione sull'application server (nel nostro caso JBoss), viene elaborato dal container. Tale elaborazione porta alla generazione di un oggetto EJB, un oggetto Home ed eventualmente di un JAR client contenente gli stub necessari ai programmi client per invocare i metodi messi a disposizione dall'EJB. Quindi il JAR client fornisce l'implementazione lato client delle interfacce home e remota dell'EJB, fornendogli dei punti di accesso alle funzionalità implementate dall'Enterprise JavaBean collocata sul server. All'atto pratico il JAR client non fa altro che inoltrare messaggi al contenitore EJB utilizzando il protocollo JAVA/RMI (Remote Method Invocation). Il descrittore di Deployment, infine, risiede in un package chiamato META-INF, e le specifiche EJB richiedono espressamente che il descrittore risieda sempre in una cartella denominata META-INF. Infine c'è da aggiungere che molti tool di sviluppo integrati (nel nostro caso Eclipse + modulo JBoss-IDE) permettono di creare in maniera semiautomatizzata e visuale i deployment descriptor XML.

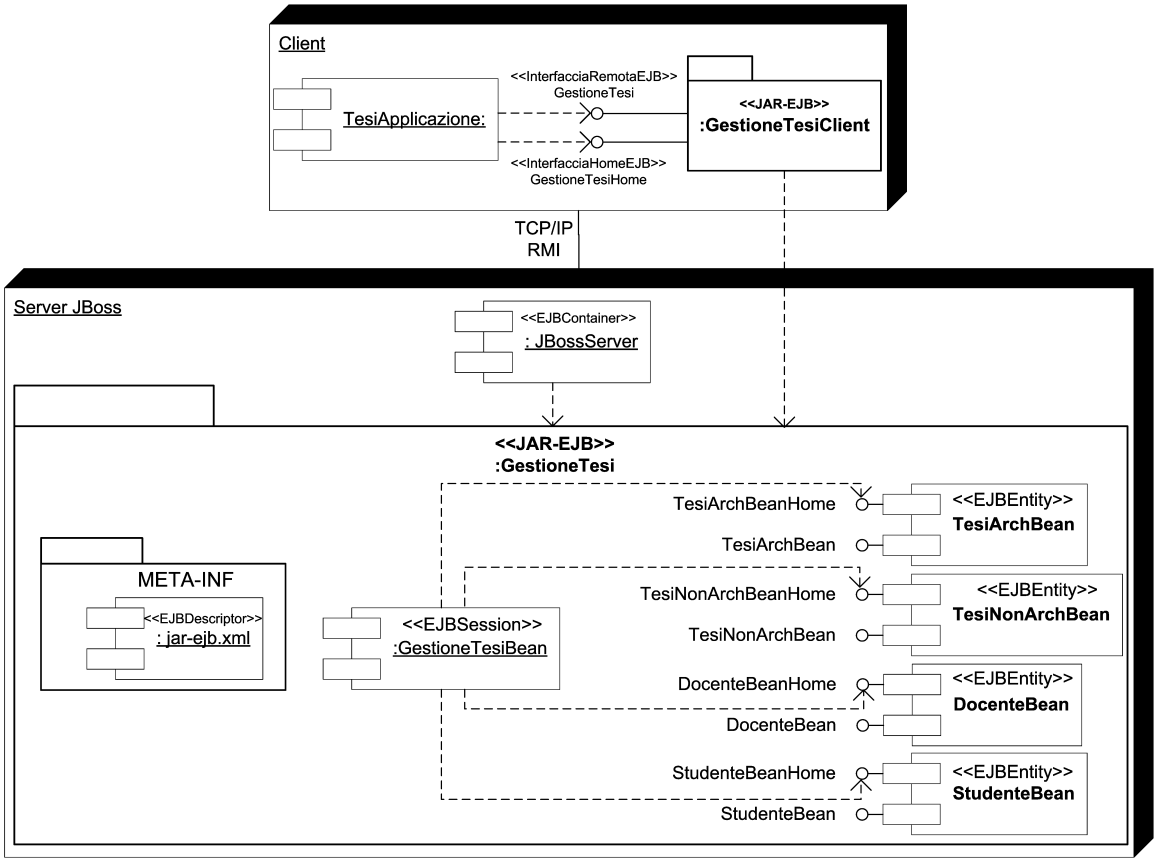


Figura 12.1: Diagramma di deployment

Parte V

Conclusioni

Capitolo 13

Considerazioni finali

Qualche considerazione in particolare va fatta per quanto riguarda il trattamento delle immagini per l'inclusione nel documento LaTeX. Abbiamo dovuto trattare immagini di diverso tipo:

- diagrammi vettoriali in bianco e nero realizzati con Microsoft Visio©;
- stampe di schermate video in formato bmp a colori;
- stampe di schermate video in formato bmp in bianco e nero.

Al fine di ottenere un documento PDF©finale di alta qualità ma che occupasse il minimo spazio possibile sono state effettuate diverse prove per valutare i vari rapporti qualità/occupazione, sia considerando l'occupazione non compressa che considerando quella compressa. Per quanto riguarda l'esportazione dei diagrammi dal formato proprietario di Visio©(.vsd) al formato Encapsulated PostScript (eps) abbiamo dovuto utilizzare tecniche di conversione composte poichè Visio©non consente di esportare direttamente in eps mantenendo una buona qualità in rapporto alle dimensioni del file. Abbiamo pertanto esportato tutti i diagrammi, in bianco e nero per occupare il minor spazio possibile, in formato bitmap ad altissima risoluzione. Inizialmente sono state esportate le immagini bitmap a risoluzioni da 100dpi a 400dpi, successivamente sono state convertite, tramite un programma di fotoritocco, in formato Encapsulated PostScript. La qualità ottenuta in rapporto alle dimensioni dei file eps è stata giudicata soddisfacente per quasi tutte le

immagini, ma una volta inserite nel documento LaTeXe convertite nel formato finale PDF©ci siamo resi conto che l'occupazione su disco di tale documento non era affatto eccessiva come ci si aspettava, inoltre la qualità di alcune immagini risultava non particolarmente accurata a causa del fattore di scala necessario alla corretta impaginazione. Il formato PDF©possiede una compressione lossless delle immagini, che senza perdita di qualità ha fatto sì che la dimensione del file non dipendesse in maniera sensibile dalla loro qualità e dalla loro occupazione su disco non compressa. Il fattore di compressione delle immagini è stato stimato mediamente attorno allo 0.01% della dimensione originale non compressa per ciascuna immagine. Si è provveduto pertanto ad una definitiva riconversione di tutti i diagrammi da Visio©in formato bitmap due colori a 600dpi, per le immagini che non sarebbero state soggette ad un alto fattore di scala oppure che non contenessero scritte con font molto piccoli, mentre per i diagrammi più grandi o complessi si è adottata una risoluzione pari a 800dpi. La successiva trasformazione in eps ha portato ad una occupazione su disco decisamente più elevata delle immagini ma una volta ottenuto il documento pdf finale quest'ultimo ha mantenuto approssimativamente la stessa dimensione del precedente documento.

Le immagini bitmap in bianco e nero sono state utilizzate per due tabelle, risultato della stampa della finestra di un programma. Sono state mantenute in bitmap, dopo aver provato ad esportarle in gif e jpg, ma la qualità non era soddisfacente: dato che tali immagini contengono prevalentemente del testo, esso presentava un effetto alias troppo evidente, causato dalla compressione con perdita di tali formati.

Le immagini a colori relative alla stampa di schermate che necessitavano di mantenere informazioni sul colore (ad esempio le immagini del diagramma a torta e di quello a nastro) sono state inserite in formato jpg in quanto soddisfacevano i requisiti di qualità minimi richiesti.

Infine va evidenziato l'effetto positivo di compressione del formato pdf: le immagini occupano in totale 90MB su disco non compresse (in formato eps e bmp) mentre il documento PDF©finale occupa appena 1,5MB.

Bibliografia

[BD] Atzeni, Ceri, Paraboschi, Torlone: *Basi di dati, seconda edizione*. McGraw-Hill, 1999.

[UML] Arlow, Neustadt: *UML e Unified Process*. McGraw-Hill, 2003.

[INGSW] Pressman: *Principi di ingegneria del software*. McGraw-Hill, 2000.

[COCOMO] *Costar 7.0 WebSite*. http://www.softstarsystem.com/demo.htm

[PPR01] Portland: *Portland Pattern Repository*. http://c2.com/ppr